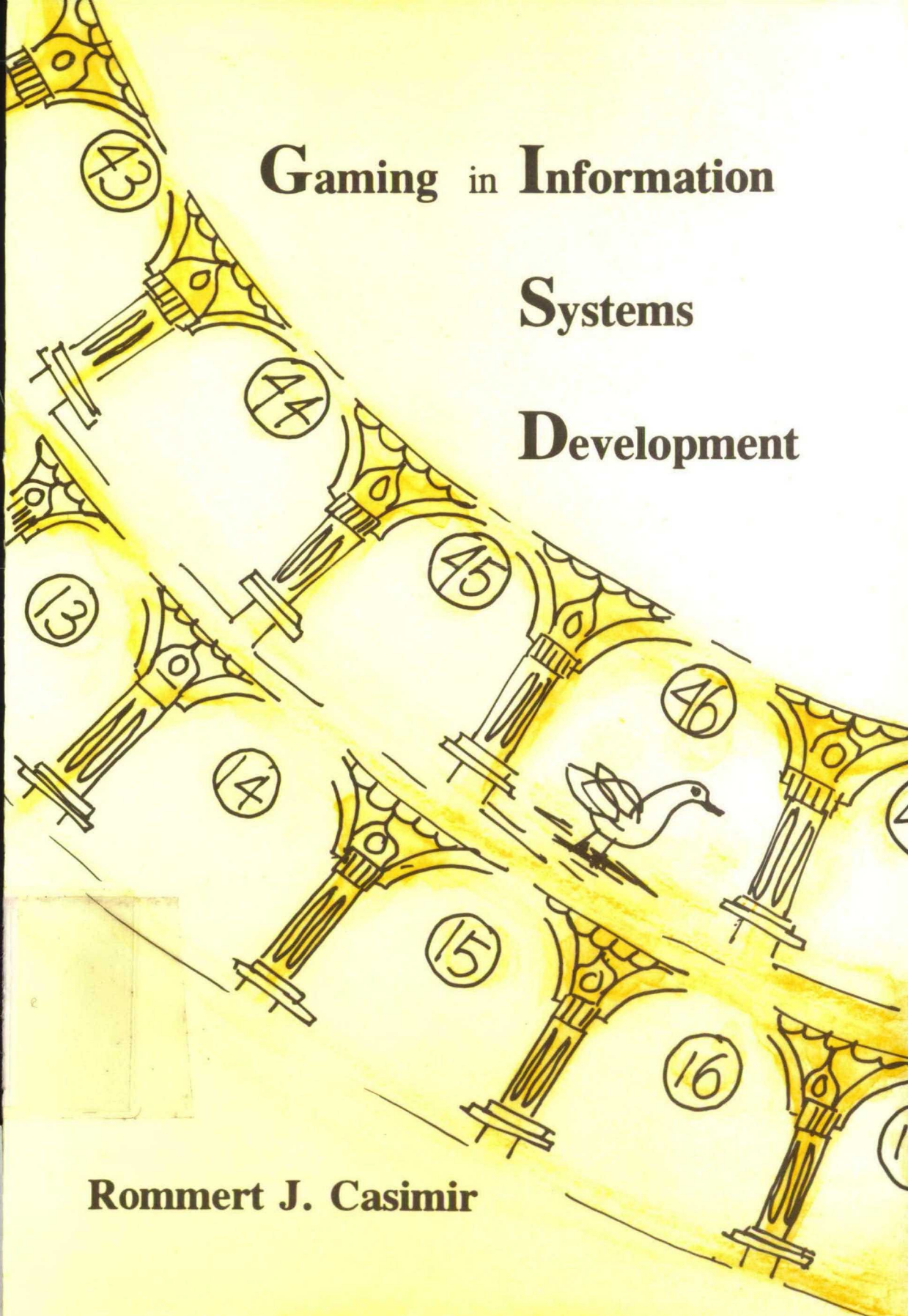
Gaming in information systems development

Casimir, Rommert Jan

Publication date:
1995

Document Version
Publisher's PDF, also known as Version of record

Gaming in Information Systems Development



Rommert J. Casimir

Gaming in Information Systems Development

Rommert J. Casimir
Tilburg University
PO Box 90153
5000LE Tilburg
casimir@kub.nl

Gaming

in

Information Systems Development

Proefschrift

ter verkrijging van de graad van doctor aan de
Katholieke Universiteit Brabant, op gezag van de rector
magnificus, prof.dr. L.F.W. de Klerk, in het openbaar te
verdedigen ten overstaan van een door het college van
dekanen aangewezen commissie in de aula van de
Universiteit op woensdag 6 december 1995 om 16.15 uur

door

Rommert Jan Casimir

geboren op 28 juni 1938 te Leiden

Promotor:
Prof. Dr. Ir. C.A.Th. Takkenberg

Omslagontwerp: Marphy Vroom-Maes

CIP-GEGEVENS KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Casimir, Rommert Jan

Gaming in information systems development / Rommert Jan
Casimir. - Goirle : Casimir. - Ill.
Proefschrift Katholieke Universiteit Brabant Tilburg. -
Met index, lit. opg. - Met samenvatting in het
Nederlands.
ISBN 90-802809-1-7
NUGI 855
Trefw.: managementspelen / informatiesysteemontwikkeling


Print: Offsetdrukkerij Ridderprint B.V., Ridderkerk

PREFACE[1]

After I completed the manuscript of this thesis, I had to do some research on the state of the art in computer science and information systems in 1970. So I found a letter to the editor from Peter Wegner in the August 1970 issue of the *Communications of the ACM*, which suggested that *perhaps the only way to continue doing good research is to deliberately write bad papers which will be rejected so as to avoid the temptations of success*. In retrospect, I have followed this advice and spent most of the past 25 years in doing the research I liked while assiduously avoiding fashionable subjects.

In 1970, my interest in management games first led me to the design of a management game for my masters thesis in business economics at the Netherlands School of Economics. When I taught computer science at Erasmus University, I explored management games as an application of cooperating processes, which in the seventies were extensively studied in the theory of operating systems.

My involvement with management games took a new turn when in 1985 I started to work on a new type of management game in a large project at Tilburg University, which aimed at increasing the use of computers in education. After two years, the project petered out, and its participants left or became regular employees of Tilburg University departments. My own part of it was continued as a regular research project, but it was stopped at the end of 1988, just after I finished the first version of Infogame (see chapter 3). One prediction that has become true was that it would take a long time before this research would bear fruit.

Two events stimulated me to finish this thesis. Aart de Zeeuw, then dean of the faculty of economics, told me bluntly there was no future for assistant professors without a doctorate, and though I could have called his bluff by pointing to the extensive safeguards in university employment contracts, I preferred to take up the challenge. More important, Cees Takkenberg agreed to supervise my thesis, and together with Martin Smits, he also inspired the adoption of Infogame in the new first year course in Information Systems, which provided the necessary experimental results. However, I would not have done all this work without the support of my wife Tine, probably the only person in the world who always believed I would finish my dissertation, and my daughters Sytske, Esther and Tanja, who heard me talk about it as long as they could speak.

---

[1] **De Nederlandse tekst van dit voorwoord en de Nederlandse samenvatting zijn te vinden op pagina 189 en volgende.**

Rolf Bijl had the supervision over the use of Infogame. He was helped by student assistants Anja Bakker, Selma van Hoorn, Michel Jongen, Frodo Lammers, Carmen Merckx, Ester Rijnders, Jeroen van Stratum, Annelies Straub and Helga van de Wijngaart. I cannot name all the students who have participated in the exercise, but their enthusiasm will be remembered. The use of Infogame in two other courses was also initiated by Rolf Bijl and Martin Smits.

In the meantime, every member of the information systems department took over part of my teaching load. I will not list anyone in particular because I do not even know how my work was divided. Cees Takkenberg lead me gently but firmly through the difficulties of interpreting and reporting my research results, and commission members Dr. J.L.A. Geurts, Dr. P.Th.M. Laagland, Dr. P.M.A. Ribbers and Dr. D.B.B. Rijsenbrij all contributed to the final version of this thesis by their suggestions for the elimination of one-sided views and obscure denotations.

The absence of acknowledgements to programmers and typists is neither an accidental omission nor a consequence of a haughty disregard for such drudgery. It only results from the fact that with present-day PC's, programming and typing no longer require assistance from a specialist. Instead, I want to thank all present and past employees of the Automation Unit of the Faculty of Economics of Tilburg University for providing me with up-to-date hardware and software, and especially, for supporting Wordperfect 5.1 for DOS throughout the time I was writing this thesis. The use of WP also permitted the compilation of an index, which includes all first, second and third authors of cited papers.

I also want to thank the employees of Tilburg University library for maintaining a splendid collection and providing an excellent indexing service. Without *Excerpta Informatica*, collecting the relevant literature, especially for chapter 2, would have been a hopeless task. Again, I will not name anyone personally because so much of the work is done quietly behind the screens.

Finally, readers from the U.S., the U.K. and other English speaking countries should be warned that this thesis is not written in their native language, but in *Broken English*, "the universal language that is spoken and understood almost everywhere" and specifically, "the language that is used by scientists at international meetings"[2]

Tilburg, 19 October 1995

---

[2] Casimir,H.B.G., "Broken English", *Scientific American*, Vol 194 (March 1956), p 96 and *Haphazard reality: half a century of science*, Harper and Row, New York, 1983, p 122.

TABLE OF CONTENTS

1 Problem formulation and research method

1.1 Problem formulation

Many methods to improve the quality of information systems have been described in the literature, but there is no generally accepted way to determine the success of those methods. In my view, one promising strategy, *laboratory research*, has been unduly neglected. The purpose of this thesis is to demonstrate how laboratory research, and more specifically gaming research, in information systems development can support and extend results from other research strategies in this field. As a prelude to the full exposition in the succeeding chapters, this chapter contains a summary description of the research domain, information systems development, an overview of research strategies, and a discussion of experimentation as a research strategy. Subsequently, the research question is formulated in more specific terms. The introduction ends with some pointers to the remainder of this thesis.

1.2 Research domain

The subject of the discipline of information systems development is the transformation of a specification of an information system that should function in the real world into a set of programs, data structures and procedure descriptions that can perform the specified function. In the literature, a difference is often made between the disciplines of *information systems*, which is mainly concerned with the organizational consequences of the design and use of information systems, and *computer science*, which primarily considers information systems design as a series of mathematical transformations. Accordingly, computer scientists are primarily employed in the design of systems with well-defined specifications, such as compilers and operating systems, and information systems specialists prefer to work on systems that help managers to solve ill-structured and ill-defined problems. The term *software engineering* is also used for the computer science approach to the

design of such systems.

Within the information systems field, a distinction is often made between *information systems development*, which covers all phases from analyzing information requirements to information system implementation, and *information systems design*, which only pertains to the logical and physical design phases. However, the distinction is not entirely clear, because the term *information systems design* is often used in a broader connotation, for example in the title of the IFIP WG 8.1 conferences on information systems *design* methodologies. The repeated use of the term information system design *and* development (Weldon 1984) adds to the confusion because design is generally considered an integral part of development. This may be due to the fact that, in everyday language, design is the process of creating and planning, and development is the gradual growth or formation of something (Collins 1987). This distinction is also applied in information systems planning, where design and development denote different strategies (Hopstaken and Kranendonk 1990). In this thesis, we use *information systems development* to describe our field of research. However, this does not imply that all information systems development phases are covered in our experiments, or that specific information systems design topics are neglected.

Information systems development is a burgeoning field, as shown by the number of books and articles in the fields of *systems design, systems development, software development and software engineering*, listed in Table 1.1:

Table 1.1 : Books and papers on information systems development[3]

Year	Books	Journal articles
1991	62	430
1992	65	433
1993	54	349
1994	28	318

[3]Excerpta Informatica Database, Tilburg University, 2 October 1995.

This body of literature is mainly concerned with the search for better methods and techniques for systems development (Friedman 1989). Hice, Turner and Caswell (1974) and Lundeberg, Goldkuell and Nilsson (1981) give no arguments for this approach, Avison and Fitzgerald (1988), Brinkkemper (1990) and Wijers (1991) cite problems with conventional methods as a motive for this quest. Blokdijk and Blokdijk (1989) assert that all methods should have a theoretical basis, which presumably means they should be deduced from accepted axioms, and Van Steenis (1990) explicitly states that his views are based on experience. However, when results of empirical research exist at all, they are treated with reverence. For example, Shneiderman et al. (1977) performed a number of experiments comparing program understanding with the use of flowcharts and pseudocode and showed that the use of flowcharts did not improve program understanding. Although the experiments involved only 60 students, the results were frequently used to argue that flowcharts were not useful, for example by Mynatt (1990). Though similar experiments were executed, the actual experiments were not replicated before Scanlan (1989) obtained quite different results.

The amount of published material on information systems is, of course, dwarfed by the amount of material associated with actual information systems, including specification texts and charts, program listings, and user manuals. There are several links between the construction of methods for information systems development and the development of information systems in practice. First, both can be characterized by the chart given in Fig 1.1, which is called the *waterfall model* when applied to the development of information systems in practice (Sommerville 1992). Nunamaker, Chen and Purdin (1990) gave a similar plan for information systems research.

The application of this model to theoretical and practical problems is given in Table 1.2. Examples are given in Fig 1.2 and Fig 1.3. It should be noted that the line between practice and theory is not entirely clear. For example, the construction of standard packages is usually considered practice rather than theory, though such programs do solve general problems.

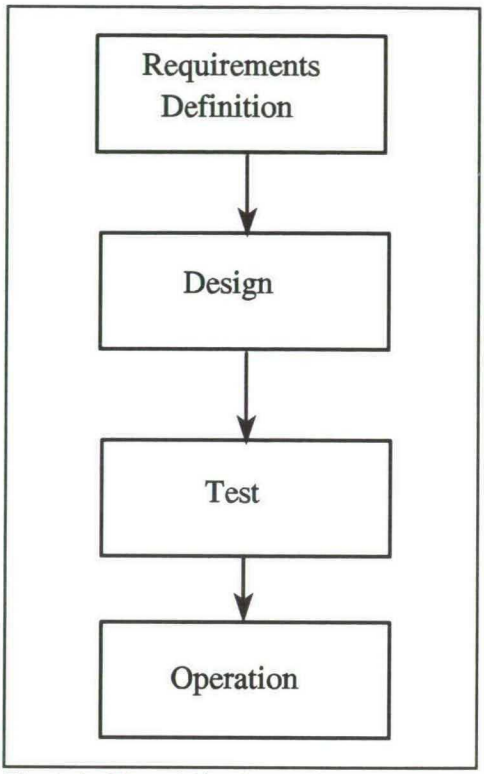


Fig 1.1: Waterfall scheme for information systems development

At what moment should a user be involved in information systems design.

Fig 1.2 : A theoretical problem

How can pig breeder Frank Priems produce a report on phosphate output required by a new law

Fig 1.3 : A practical problem

Table 1.2: Waterfall model for theoretical and practical problems.

Stage	Theory	Practice
Requirements definition	Description of a general problem	Description of a specific problem.
Design	Definition of a theory or method	Design of an information system
Test	Laboratory test of theory or method	Prototype test, program unit test or system test
Operation	Use of theory or method in actual design of information systems	Use of information system

A second link between information systems theory and practice is that the results of information systems theory are used in the actual design of information systems. Conversely, information systems are the object of research of information systems theory. Finally, designers often double as researchers. For example, some prominent computer scientists, such as Turing award winners Dijkstra (1972), Wirth (1985), Ritchie (1984) and Thompson (1984) have started their careers as designers of successful systems (Dijkstra 1968; Wirth 1970; Ritchie and Thompson 1974).

1.3 Research strategies

Several classifications for research methods or strategies have been proposed. Usually, a distinction is made between *empirical research* and other types of research. Van Horn (1973) classified empirical research into *case studies, field studies, field tests* and *laboratory studies* (Hamilton and Ives 1982). Benbasat (1985) called field tests and laboratory studies *field experiments* and *laboratory experiments* respectively. Whereas the first change is only a question of terminology, the second change limits laboratory studies to studies with experimental control. Cooper (1988), Galliers (1985) and Scott Morton (1985) call field studies *surveys* whereas Benbasat (1985) and Cheon, Grover and Sabherwal (1993) distinguish field studies and surveys as separate research strategies. Following Christensen (1988), we consider case studies and surveys as special forms of field studies. Two other forms, *participant observation* and *meta analysis* are not mentioned in the information systems literature. The three remaining types of empirical research are characterized by two parameters, the *degree of experimental control* and the *type of setting*. The result is given in Table 1.3.

Table 1.3: Research Strategies

	Natural setting	Contrived setting
Experimental control	Field experiment	Laboratory experiment
No experimental control	Field study	Laboratory study

By completing the table we find a research strategy that has been overlooked in the literature, the *laboratory study* without experimental control. Nevertheless, it has been used frequently. For example, animal behaviour has been studied in zoos, and human behaviour in playgrounds and classrooms. Though we originally planned at least one controlled experiment, the research reported in chapter 5 actually contains only laboratory studies. Because the term *experimental research* is often restricted to research with experimental control, the term *laboratory research* is used for our experiments.

In the context of information systems research, laboratory research always pertains to testing human reactions. It should be distinguished from the experimental work in computer science that is usually performed in *computer laboratories*, such as testing the efficiency of new algorithms for parallel computers.

An important method in laboratory research is *gaming*. Typically, gaming experiments simulate a fairly complex and realistic environment, which induces a relatively natural behaviour from participants. A more precise definition of games and a classification of gaming are given in chapter 3.

The role of the computer in laboratory research is diverse. It is the subject of research in the field of human-computer interaction, whereas in gaming, it is used to simulate a complex environment, for example, a market with many participants.

There is less unanimity on the classification of nonempirical research. Christensen (1988) reserves the label *science* for empirical research. In his view, other ways to acquire knowledge, such as *intuition* or *rationalism*, which denotes reasoning from given premises, are nonscientific. It should be remarked that Klein and Lyytinen (1985) challenge this restricted interpretation of science, but a discussion about the essence of science is outside the scope of this thesis. Hamilton and Ives (1982) divide nonempirical research into two classes: (i) *conceptual* and (ii) *tutorial, review and other*. Vogel and Wetherbe (1984) distinguish *theorem proof*, *engineering* and *subjective argumentative*. This classification was subsequently used in surveys by Cooper (1988), Teng and Galletta (1990) and Vogel and Nunamaker (1990). Alavi and Carlson (1992) divide nonempirical research into three classes: *conceptual orientation, illustrative,* and *applied concepts*. However, the third class was found to be quantitatively insignificant. Conceptual orientation seems to be equal

to Vogel and Wetherbe's theorem proof, and illustrative contains both engineering and subjective argumentative. Nunamaker, Chen and Purdin (1990) distinguish *theory building*, *systems development*, *experimentation*, and *observation*. Empirical research is represented by observation (field studies) and experimentation (field experiments and laboratory experiments). Theory building encompasses theorem proof and subjective argumentative in the Vogel and Wetherbe classification, and systems development is equivalent to engineering.

We follow the Nunamaker, Chen and Purdin classification, but we prefer the terms conceptualization instead of theory building and engineering instead of systems development. This classification can be linked to the stages in the life cycle of a theory or an information system described in Table 1.2. For each stage, the preferred research strategy is listed in Table 1.4[4]

Table 1.4: Stage of development and research strategy.

Stage	Preferred research strategy
Requirements definition	Conceptualization
Design	Engineering
Test	Experimentation
Operation	Observation

Whereas engineering, experimentation and observation are relatively well defined, conceptualization covers a broad spectrum, ranging from the explanation of speculative theories to precise mathematical derivations. In this respect, the field of information systems differs from disciplines that predominantly rely on a single method, such as operations research, where the exclusive use of mathematical theorem proving as a method is commonplace, though its appropriateness was questioned by Daellenbach (1994). Finding a generally acceptable classification within the field of conceptualization is, however, outside the scope of this paper.

[4]Though all premises are given by Nunamaker, Chen and Purdin (1990), Table 1.4 is not explicitly represented there. Presumably, this was left to the reader as an exercise.

1.4 Empirical research

Though the importance of empirical research is growing in some parts of information systems, notably management support systems, conceptualization and engineering research dominate the field of information systems development as a whole, especially the part of the discipline normally included in computer science. For example, Pfleeger (1995) treated experimentation in software engineering as an entirely new field without any reference to the existing literature on experimental research on information systems.

Four arguments for the neglect of laboratory research can be advanced:

a) The strength of research strategies used in preceding stages, i.e. conceptualization and engineering.
b) The advantage of bypassing the test stage by a direct transition from design to operation.
c) The inadequate validity of laboratory experiments.
d) The lack of appropriate tools.

The strength of conceptualization and engineering

Many computer scientists, for example Hoare (1969) and Gries (1991), assert that the correctness of a program can be deduced from its specification by mathematical induction, and that, consequently, it is not necessary to test a program. This notion is, however, rejected by some computer scientists (Tanenbaum 1976), and it is even less valid for information systems development, where specifications are not defined exactly. However, many authors insist on the importance of formal modelling. A typical example of a statement that is assumed to be true without empirical proof is the basic premise of many system development methodologies that systems development should be partitioned into stages with explicitly defined milestones.

The advantage of bypassing the test stage

Experiments in physics and medicine are justified by the high cost of introducing a product without laboratory testing, because the producer will be held responsible for damages from an untested product. On the other hand, in many industries it is assumed that the market will choose the best product. For example, the market has proven the superiority of Coca Cola and Macdonald hamburgers over local produce in many parts of the world. In a similar vein, it could be said that the market should weed out unsuccessful methods for information systems development. Moreover, it may be advantageous to speed up the introduction of new methods because of the urgent demand for new methods and techniques that can cure the persistent *software crisis*. However, the high cost of information systems is a recurrent theme in the literature, and many new methods have not lived up to the claims of their designers. Consequently, laboratory research that assesses whether a new method really decreases costs may be well worth the expense.

The inadequate validity of laboratory experiments.

Doubt of validity of laboratory research especially concerns the widespread use of students as players (Hughes and Gibson 1991). However, the fact that students react differently from managers, or that results from field studies differ from laboratory results (Dennis, Nunamaker and Vogel 1990) does not invalidate the results of laboratory research. It is adequate if success of an information system method in the laboratory is positively correlated with success in the field. In medical research, new treatments that are harmful to rats are not dispensed to humans, although they might be quite harmless in view of the difference between rat and man. Another argument against laboratory experiments was suggested by Galliers (1993), who argues that they are inappropriate for studying the organizational aspects of information systems. However, he notes the value of gaming in this area, and in our view, gaming is a special form of laboratory experimentation.

The lack of appropriate tools.

Albeit it is easy to set up simple experiments in man-machine interaction, for example to investigate the effect of graphical and tabular representation of data, setting up gaming experiments is fairly difficult. A gaming experiment can either use an existing game or a specially built one. Though the cost of selection and implementation of an existing game should not be underestimated, using an existing game is certainly cheaper than building a new one, but it has two drawbacks. First, no game may be suitable for the task, and second, if a commercially available game becomes too popular, results will be distorted because some players may know the game beforehand. For example, a Chess tournament is not suitable as an intelligence test for a random group of students.

In my view, existing management games cannot be used for experiments in information systems development because the environment simulated in such games is too simple to expose the difference between players provided with superior and inferior information systems. This was the main motivation to build Infogame, a management game tailored to laboratory research in information systems development. Moreover, the design of a new game offered the challenge to explore the theoretical foundations of decision making in organizations (Casimir 1986). Consequently, building the game was considered part of the research endeavour. The effort in designing, building, testing and tuning Infogame was significant, and the description of the game in chapter 4 is an important part of this thesis.

1.5 Research questions

As stated in section 1.1, the central theme of this thesis is:

What are the advantages and limitations of laboratory research in the study of information system development methods and tools?

The advantages of laboratory research could be established in a single blow by devising a new hypothesis, proving it with a laboratory experiment, and

subsequently validating it by empirical research. This is fairly difficult, because the new hypothesis should be accepted as an important issue by theoretical and empirical researchers in the field. Accordingly, this route is only open to an experimenter who is also a first-rate theoretician and an accomplished empirical researcher.

Consequently, we use the step-wise approach illustrated in Fig 1.4.

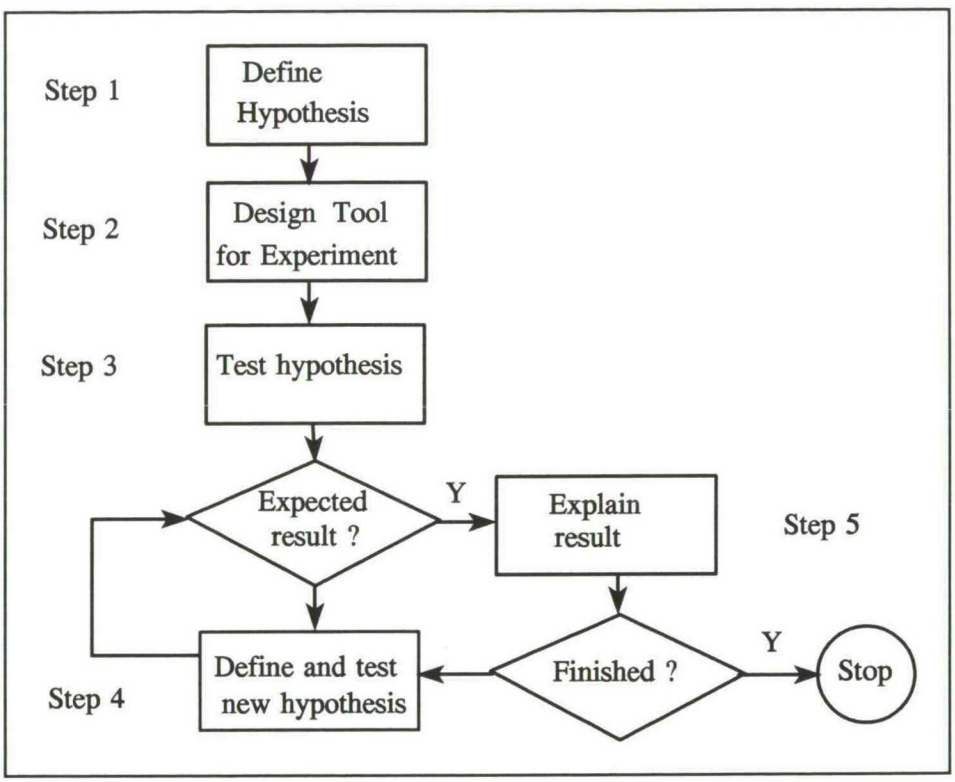


Fig. 1.4: Stepwise approach to testing

In step 1, we devise a hypothesis that is generally accepted and of sufficient interest. The precise definition of this hypothesis and its place in information systems development theory will be given in chapter 2, but in general terms it is stated as:

> *The quality of successive milestones in the system development life cycle is positive correlated*

In step 2, we create a laboratory environment and in step 3, we design an

experiment to test this hypothesis. When the experiment delivers the expected results, our trust in the laboratory is increased, and we proceed with step 4 to use it to test a more controversial hypothesis. Eventually, laboratory results will contradict established wisdom. This will cause a reexamination of existing theories in step 5, and the corroboration of the unexpected results by other research strategies will signal a real success of laboratory research.

As shown in chapter 5, the research reported in this thesis only covers the first four steps. Experiments on controversial hypotheses delivered no results that were clear enough to instigate a reexamination of existing theory.

When the number of experiments increases, the specific subfields of information systems development that are suitable or unsuitable for laboratory research can be determined. This will offer new opportunities for laboratory research, some of which are discussed in chapter 6.

Our research strategy differs from the strategy used in some other gaming studies. Gremmen (1989) and Coppieters (1990) focused on step 2 by designing a new game. On the other hand, step 2 was curtailed by Van Schaik (1988), who used an existing game, and Vennix (1990), who incorporated an existing computer simulation model in his game. Woltjer (1995) covered all steps by designing a new game, testing its educational effectiveness, and exploring its use in economics research.

1.6 Outline of this thesis

Three research strategies are used in this thesis. Chapters 1,2 and 3 are conceptual in nature, and they are primarily based on the existing literature. Moreover, these chapters ascertain that the contents of chapters 4 and 5 are original by a broad survey of the literature where references to this type of work should appear.

Chapter 2 discusses the discipline of information systems development. It focuses on the results of existing theories, which have been developed in the conceptual tradition. Section 2.2 focuses on quality assessment of information systems, and section 2.3 describes information systems development methods.

Chapter 3 contains a discussion of gaming in section 3.2, an overview of laboratory research in information systems in section 3.3, and a synthesis of both subjects in section 3.4. Readers are warned that some of the material is treated in a scholarly, rather than a scientific manner.

Chapter 4 is in the engineering tradition. It contains a description of Infogame, the game that has been developed for this study. The main emphasis is on design decisions. From the description it should be clear that the research described in chapter 5 could not be done with a standard management game. Chapter 4 also addresses prospective users of Infogame. Its use in education and research for laboratory experiments that are not feasible with conventional management games is welcomed because of the advantages of using a common gaming environment (Courtney, DeSanctis and Kasper 1983).

Chapter 5 is empirical; it describes the results of experiments with Infogame. If laboratory research was commonplace in information systems development, chapters 3 and 4 would be superfluous. If information systems development was a *normal science* in the sense used by Kuhn (1970), chapter 2 could be replaced by a short reference to a textbook, and chapter 5 would constitute a complete research project.

Chapter 6 draws conclusions and contains suggestions for further research. Moreover, it addresses the question what topics in information systems development are suitable for laboratory research in general and research with Infogame in particular.

The order in which chapters and sections may be read is described in Fig 1.5. Each block shown in Fig. 1.5 can be read after all preceding blocks in the graph. However, some sections may be skipped because they only provide background information.

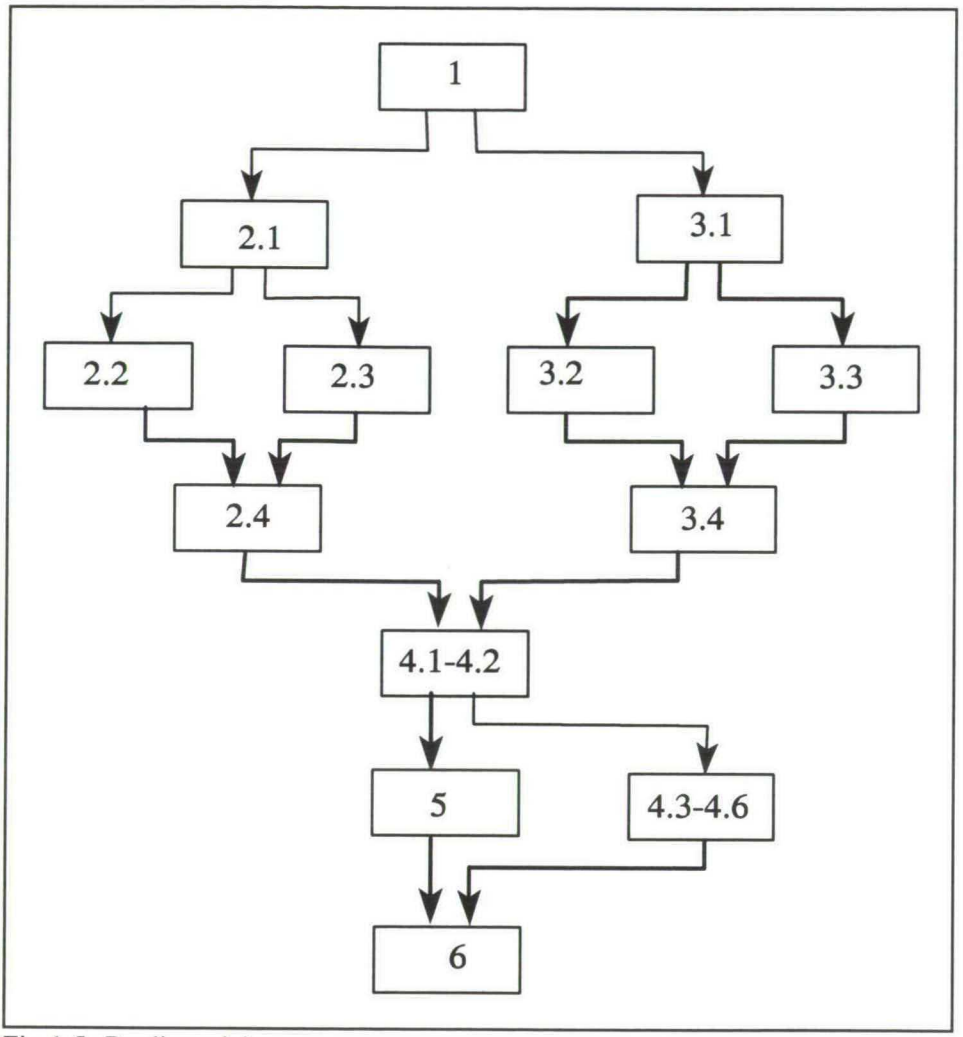


Fig 1.5: Reading advice

2 Information systems

2.1 Introduction

The purpose of this chapter is to define one or more hypotheses on information systems development that can be tested in a laboratory. Such hypotheses are not readily found in the existing literature. On the one hand, a general hypothesis such as *Use of Decision Support Systems improves decision making* may be too difficult to prove (Van Schaik 1988), on the other hand, a hypothesis which concerns only a restricted environment, such as *In a BAP study, commitment of higher-level management can be gained by presenting to the executive management the objectives, advantages, expected outputs and resource requirements of the systems planning study* (Laagland 1983) may not be of sufficiently general interest.

The search for hypotheses is troubled by the lack of a single paradigm in information systems development. For example, Hirschheim and Klein (1989) distinguished four paradigms for information systems development, and they still did not cover all possible views. Accordingly, we will delineate our approach by presenting a classification of views on information systems development and selecting our preferred view. This classification is based on the notion that authors on information systems disagree on three basic issues. They may hold a *broad* or *narrow view* on information systems development, they may see information systems development as an *engineering discipline* or a *science*, and they may consider the study of information systems as a *social science* or a *part of mathematics*.

Broad or narrow view

In the broad view, information systems development concerns the development of manual as well as automatic information systems, or in the words of Verrijn Stuart (1989), *the totality of all formal and informal data representation and processing activity within an organization*. For example, when an accounts receivable system is analyzed, the alternative of asking cash payment is considered as well. In

the narrow view, the information systems discipline is only concerned with automating existing applications. Ostensibly, the broad view is held with respect to decision support systems and strategic information systems, and the narrow view with respect to clerical systems. This can be deduced from the statement by Turban (1993) that: *About $ 30 billions of DP funds in the USA goes to support $450 billion of management / professional activities while $50 billion of DP dollars go to support $150 billion of clerical activities*, and from the widespread view that the gains from transaction processing systems are equal to the saving in clerical costs (Kleijnen 1980). In the broad view, clerical operations are part of information systems, and consequently, the extent of the clerical activities themselves should be the subject of information systems research.

Engineering discipline or science

When information systems development is seen as an engineering discipline, the task of the researcher is to produce information systems or methods for information systems development. He is working with his eyes on the future, and only looks at the past for clues to improvements. When it is seen as a science, the analysis of existing systems is an aim in itself. Engineers and scientists typically differ in their appraisal of failures. The engineer will attribute them primarily to incompetence of the persons involved, and show how to prevent them in the future, whereas the scientist will first try to explain them. As Friedman (1989) noted in one of the few books that is an exception to this trend, the information systems development field is mainly seen as an engineering discipline.

Social science or part of mathematics

Because information systems interact with people, information systems development should be considered a social science. However, because the social sciences are low in the scientific pecking order, this has been consistently denied. To this end, a neat stratagem has been borrowed from economics and management science: instead of information systems proper, models of information systems are

studied. For a problem that is formulated within the scope of a model, any mathematically correct solution will be accepted by the research community. For example, in the newsboy problem (Lapin 1988; Shogan 1988), it can be formally proven that a newsboy with perfect information will attain higher profits than a newsboy with imperfect information. No serious operations researcher will think it necessary to corroborate such results by interviewing newsboys in the field. The mathematical view has dominated computer science from its inception and is now gaining importance in the information systems development field. For example, in the proceedings of the 1989 IFIP TC 8 /WG 8.1 Working Conference on Information Systems Concepts (Falkenberg and Lindgreen 1989), 11 of the 13 papers took a modelling approach, and none of those contained a reference to any article in core MIS journals such as MIS Quarterly, Information and Management or Journal of MIS, which are more oriented towards the social science approach to information systems.

When information systems development is seen as a social science, views on methods will be coloured by political views. This is acknowledged by Hirschheim and Klein (1989), who classify views along the axes *order - conflict* and *subjective - objective*. Overtly political views are expressed by Kraft (1977) and Greenbaum (1979), who see information systems development methods as instruments to deskill programmers. Robey, Smith and Visijayasarathy (1993) concentrated on the political process of information systems development. In this thesis, the political side of information systems development will be ignored. First, it cannot be discussed without a thorough exposition of political theories, second, it would take a substantial effort to unite the many existing models, and third, some political themes, such as hidden agendas (Van Steenis 1990), are only superficially discussed in the literature.

The split between the mathematical and the social science view is the most pervasive distinction in information systems. It is also called the difference between computer science and information systems, between hard and soft science or between dry and wet culture (Goguen 1992). For the sake of courtesy, some other terms for the two cultures will not be cited here.

A combination of the three characteristics results in eight distinct paradigms, that, however, cannot always be uniquely associated with an established discipline. Characteristics and disciplines are listed in Table 2.1.[5]

Table 2.1 Approaches to information systems

	Broad		Narrow	
	Engineering	Science	Engineering	Science
Social Science	Organization Design	Broad MIS Science	Information Systems Development	Narrow MIS Science
Mathematics	Management Science		Software Engineering	Computer Science

A similar classification was given in a review of classifications of the MIS discipline by Holsapple et al. (1994). They divided research along two axes, *actions - ideas* and *human - computer*, which correspond to our *engineering - science* and *social science - mathematics* dichotomies respectively. However, because of the introduction of other criteria such as research strategy, and the shift of the classifications with respect to the axes, the resulting classes do not correspond to the classes defined in Table 2.1.

The research in this thesis belongs to broad MIS science, but the hypotheses to be proven will be gleaned from all fields in Table 2.1. Special attention will be given to hypotheses that are current in more than one field. We start from a theme that has already been mentioned by Nolan (1973): *Computer applications for middle-to-top-management have almost always been plagued with incidents of user dissatisfaction.* This lament has been echoed in a myriad of publications, but it has not been substantiated by a solid body of empirical research. For example, Yourdon (1992) states that 25% of large system development projects never finishes and attributes this statement to DeMarco, but DeMarco (1982) writes that 15% of projects never deliver anything, without giving any further reference or empirical

[5] Table 2 is new, but the literature was not extensively searched to prove its originality.

data. Siskens, Heemstra and Van de Stelt (1989) ascribe the magic 25% to an earlier book by Yourdon and to C. Jones. In one of the few published empirical studies (Jenkins, Naumann and Wetherbe 1984), it was found that users where satisfied with 72% of projects. However, this study only included projects completed during the preceding two years that were selected in cooperation with the systems development manager, and user satisfaction was estimated by the project manager. Empirical research is further hampered by the scarcity of data on information systems development recorded by organizations. In their empirical study on system development cost control, Siskens, Heemstra and Van de Stelt found that only 16% of respondents maintained the data necessary to accurately analyze cost overruns and late deliveries, the other major concern in the information systems development field.

The notion that information systems often fail to serve user needs has been the starting point for two lines of inquiry: *How to determine the quality of information systems*, a subject in the MIS Science field, and *How to build quality information systems*, the main topic in the information systems development field. Most articles concentrate on one of the two themes, which we call *assessment research* and *engineering research* respectively. Assessment research papers usually contain a detailed description of quality metrics, but they tend to propose solutions only in general terms. On the other hand, engineering research papers address quality determination only in vague terms, but they habitually contain detailed descriptions of methods to improve system quality. In the words of Brooks (1987), the latter category often is in search of a *silver bullet*, a magical cure against all ills that bedevil software development. He implies that such a search is characteristic of the pre-scientific stage of a field. In spite of this warning, the search for silver bullets has continued (Cox 1990). Nevertheless, though the belief that a single method will end all problems may not be warranted, new methods and techniques proposed in engineering research articles may well have real though restricted advantages.

2.2 Assessment research

2.2.1 Introduction

The elusive character of the notion of quality is best shown in the words of Juran (1988, p 4): "there is no known short definition that results in a real agreement on what is meant by quality". However, on p 5, he actually gives such a short definition, *fitness for user*, but proceeds by identifying the complications resulting from the existence of multiple users. The variety of meanings is also apparent from the list of meanings of *quality* in the second edition of the Quality Control Handbook (Bingham 1962). According to Juran's definition, quality can only be determined by the user, and it can only be resolved after its use or *ex post*. The user is the person or organization for whom the product is ultimately intended, which is not necessarily the person who directly uses the product. For example, the attire of an actress is not intended to please herself, but to please the public. If a product fails to satisfy its users, the causes for this failure may be sought to ensure this will not happen another time. In this way, the quality of the *process* that delivers the product is enhanced by feedback. When a process is performed repeatedly, the quality of its products can be accurately predicted, which is equivalent to quality determination *ex ante*. Focusing on the factors that sustain quality is called the *causal* approach to quality, whereas stressing the quality of the product for the user is called the *teleologic* approach to quality (Van der Pijl 1993). A similar distinction is that made between the *behaviourial quality*, which can be assessed by the user, and the *structural quality*, which can only be determined by analyzing the product (Wesselius 1993). The difference between the two classifications is that the causal approach excludes considerations of the purpose of the product, whereas the structural approach does not. Table 2.2 shows the classification resulting from the combination of the ex ante/ex post and causal/teleologic criteria.

Table 2.2 Approaches to information systems quality.

	Causal	Teleologic
Ex ante	Software quality	User requirements
Ex post	Success factors	User satisfaction

The fields in Table 2.2 can also be distinguished by the research strategies applied. Literature on software quality is mainly prescriptive, papers on user requirements tend to be speculative or theoretical, and success factors and user satisfaction are appropriate subjects for empirical studies. In the literature, the prescriptive approach to software quality and the empirical approach to user satisfaction overshadow research on user requirements and success factors. Accordingly, section 2.2.2 on teleologic approaches stresses the ex post view, and section 2.2.3, which covers the causal approaches, emphasizes the ex ante view.

2.2.2 Teleologic approaches

The main problem in the teleologic approach is to define the notion of information system success. The yardstick for information system success has also been called the dependent variable, whereas the factors determining product quality have been named the independent variables. In a survey of 100 empirical research papers, DeLone and McLean (1992) presented six dependent variables, including *use, user satisfaction, individual impact* and *organizational impact*. User satisfaction was chosen as a dependent variable in 28 field and 5 laboratory studies. Measures to rate user satisfaction ranged from a single question to the 39-item Bailey and Pearson (1983) questionnaire. It should be noted, however, that the Bailey and Pearson instrument also includes causal aspects, because satisfaction is determined for separate information systems attributes, such as documentation. On the other hand, individual impact was evaluated in 24 laboratory studies and 14 field studies. Apparently, in a laboratory, impact can be measured more easily than in the field.

Of course, the ultimate aim of information systems should be to support the success of the organization (organizational impact) or the individual working in the

organization (individual impact). Nolan and Seward (1974) introduced user satisfaction as a surrogate for impact, because it could be measured more easily. Marsden and Pingry (1988) and Melone (1990) opposed its use on theoretical grounds and Galletta and Lederer (1989) advanced some cautions because an experiment challenged the reliability of a user satisfaction test. On the other hand, an empirical assessment by Gatian (1994) showed a high correlation between user satisfaction and organization success. A theoretical argument for the application of user satisfaction as a success indicator is the lack of reports on anomalies, i.e. highly acclaimed information systems leading to organization failures or much criticized systems contributing to organizational successes.

For our research, the main value of the literature that applies the teleologic approach is its support for user satisfaction as a dependent variable.

2.2.3 Causal approaches

2.2.3.1 Background

When the causal approach is used, the factors contributing to information systems quality are often studied without systematic research into the relation between those factors and a metric for information systems quality, such as user satisfaction. Four reasons for this deviation from normal scientific practice are:

1 The task of improving software quality may be too urgent to await the results of scientific inquiry. Software engineers may compare themselves to Florence Nightingale, who, on arriving in the field hospital in the Crimea, started with improvements without waiting for the results of research on patient satisfaction or the causes of illnesses (Underwood 1974).

2 Books and articles written by experienced practitioners may be based on data that are not available for scientific research. For example, when Edward Yourdon (1976) states: *I 've had the opportunity to visit literally hundreds of organizations - and everyone is having the same problem*, there

are ample reasons to believe his views rather than the survey results from a Ph.D. student.

3 The user may be wrong. Many producers of quality products, such as chateau owners and computer science professors, think they must educate their customers rather than cater to their uncouth tastes. This attitude may be more common in Europe than in North America, which partly explains a difference in the emphasis in the literature.

4 Agreement on the quality of an information system may be more important than the quality of the information system itself. In this view, information systems developers and users can be compared to a party lost in the midst of the desert, where persisting in the chosen direction may be more important than the choice of direction. This aspect of information systems quality is stressed in the literature on Group Decision Support Systems.

2.2.3.2 Scoring models

In the causal approach, a scoring model is often used to determine quality. In an *additive* scoring model, total quality is determined by the formula:

$$q = \sum_{i=1}^{n} q_i w_i$$

q system quality

q_i quality of attribute i

w_i weight of attribute i

Attributes should be mutually independent. Otherwise, the importance of the independent attributes is underrated.

In a *multiplicative* scoring model, quality is determined by:

$$q=\prod_{i=1}^{n} q_i^{w_i}$$

A very simple scoring model is the *check list*. When a check list is used, it is assumed that quality is sufficient if all requirements on the checklist are satisfied. Accordingly, a checklist is a multiplicative scoring model defined by:

$$q=\prod_{i=1}^{n} q_i \ , q_i \in \{0,1\}$$

In an additive model, a small number of attributes suffices, because adding an attribute with a small weight is ineffective. In a multiplicative model, the list of attributes may grow indefinitely, because a zero value of any attribute can reduce the total score to zero.

Scoring models have been used in the selection of research and engineering projects since the fifties. Early applications were described by Mottley and Newton (1959) and Hertz and Carlson (1963), who both used a multiplicative model, and Dean and Nishry (1965), who applied an additive model. A survey of 34 models, mainly from English-language sources, was compiled by Dreyer (1974). This inspired Bemelmans (1984) to use a similar model to determine the quality of information systems. He has been followed by Delen and Rijsenbrij (1990a; 1990b), whose model for quality determination has been extensively used in practice in the Netherlands, for example in a number of masters theses at Tilburg University. Earlier, an additive scoring model was used for information systems selection by Gilb (1976). Another popular model was proposed by Bailey and Pearson (1983), who went back to the literature on job satisfaction (Wanous and Lawler 1972). Though scoring models are used frequently, Eilon (1993) criticized them as mere calculation exercises without a foundation in reality, thereby faintly echoing Erasmus'(1515) scorn for the exact computation of the number of days a sinner had to spend in purgatory. However, when the use of scoring models in different

disciplines is studied, it is plausible that they are predominantly used during the transition from the use of purely qualitative methods to the use of more sophisticated quantitative methods.

Scoring models can be used either as audit tools or as prescriptions. In an audit, the overall error is minimized if the product of weight and average error is constant for all attributes. Consequently, the weight of an attribute should be known to the valuator to enable him to spend more time in assessing important attributes. A number of proprietary audit tools have been developed by EDP auditing firms. An audit may result in an advice to improve on some attributes of information systems quality, or it may be used as a *quick scan* to identify attributes that should be studied in more detail (Swinkels and Brouwers 1990). The use of a check list is a traditional application of a scoring model as a prescription because it is obvious that the quality of each attribute on the check list should be sufficient. For example, Boehm (1983) gives a simple prescription with a check list containing only seven "principles".

For our research, the literature in the causal tradition will mainly be used as a source of subjects for further research. For example, the relation between information systems quality and any of the 41 quality attributes mentioned by Delen and Rijsenbrij (1990a), such as proficiency of system designers or correctness of data, may be empirically tested. We will study four interrelated principles listed by Boehm, *Manage with a phased life-cycle plan* with *continuous validation*, *disciplined product control* and *clear accountability*.

2.3 Engineering research

2.3.1 Introduction

According to Friedman (1989), the literature on information systems development is mainly prescriptive. In his view, the prescriptive literature gives no accurate view of current systems development practice, but nevertheless, it is important because it influences practice. For example, the prescriptions in a book on systems development written in 1974 may be extensively implemented by 1984. Accordingly, the importance of books and articles should be measured by the

number of sales and copies made, and not by references in scientific journals.

Another link between prescriptive literature and the real world is the likelihood that the tools and methods that are recommended were available when the book or article was published. In this respect, a distinction must be made between books and articles sketching broad views of information systems, that are primarily directed at senior management, and detailed prescriptions for information systems specialists. The first group suffers most from what Friedman calls *the problem of tenses*, which means that technologies in the laboratory stage are presented as generally available. For example, in 1965 top managers may have been made to believe that five years hence, all middle management decisions would have been automated, and in 1985, the same could be suggested for artificial intelligence.

The prescriptive literature may also elucidate the state of the real world at the time of writing because practices that are condemned were probably widespread. This method has been popularized by Elias (1969), who determined the advancement of civilization in the Middle Ages from books on good manners. An example from information systems is the rule that programmers should not enter the computer room, which became widespread in the late sixties (Greenbaum 1979).

The descriptive literature on information systems development methods is scarcer than the prescriptive literature, at least in the archival journals. Moreover, descriptions of the actual use of a method are only published well after its introduction. Accordingly, such papers are only beginning to appear in the eighties, for example in the work of Guimaraes (1983; 1985). More recently, Avgerou and Cornford (1993) stated that the methodologies movement did not address the question whether particular methodical practices resulted in better outcomes.

2.3.2 A sketch of the history

In this section, only a rough sketch of the history of information systems development will be given. It is based on Friedman's (1989) comprehensive account and on selected items from the contemporary literature, and it is coloured by personal reminiscences. Though McCracken, Weiss and Lee (1959) already proposed a systematic approach to systems development, and Laden and Gildersleeve (1963)

gave a fairly elaborate prescription, the first full-fledged method descriptions date from the late sixties. Examples are the first three volumes of the Handbook of data processing management (Rubin 1970), the ARDI handbook (Hartman, Matthes and Proeme 1968) and the NCC's Systems Analysis Package (Daniels and Yeates 1969). The NCC method appears to have changed markedly around 1970, because the 1969 edition featured a functional decomposition, whereas the 1971 edition described the more familiar division in phases already known from ARDI. Benjamin (1971) gave a comparison of several methods that were in existence at the time. It should also be noted that he advocated the use of prototyping.

The systems development method, SDM, (Hice, Turner and Caswell 1974) has become the standard project management method in the Netherlands, mainly because it was promoted by leading software houses such as Pandata and Volmac. As a project management tool, SDM allows the use of externally acquired methods for data and process description. An example of a method for process description that became popular in the late seventies is ISAC (Lundeberg, Goldkuell and Nilsson 1981). The notion that data are the prime information systems resource led to the increased use of data modelling methods, such as the Entity-Relationship model (Chen 1976; Teory, Yang and Fry 1986) and NIAM (De Troyer 1993).

In Europe, the use of methods really caught on in the early eighties. For example, the leading Dutch information systems journal "Informatie" carried a series with descriptions of 24 systems development methods from February 1981 through March 1982. In the UK, SSADM (Downs, Clare and Coe 1988; Hares 1994), developed by the CCTA, became the standard method[6]. In contrast to SDM, SSADM includes data and process modelling techniques in addition to project management methods.

In the USA, the emphasis was different. Though the phased approach, known as the waterfall model (Boehm 1981; Sommerville 1992) or the System Development Life Cycle (SDLC) approach, is widely applied and the importance of

---

[6] Though the NCC is involved in the development and administration of SSADM, there is no evidence of a direct link to its earlier efforts in Information System Development methods. It is also an intriguing question why government influence on Information Systems Development should be so much larger in the UK than in the Netherlands during a period when the UK government was much more committed to free enterprise.

milestones was already stressed by Royce (1970), reviews of methods discuss a mixture of techniques rather than focusing on methods covering the full SDLC (Sumner 1986; Necco, Gordon and Tsai 1987; Palvia and Nosek 1993). For example, in the seventies, so much interest was given to structured design in general and structured programming in particular (Canning 1974; Yourdon 1976), that, together with the chief programmer team (Baker 1972), it was singled out as the equivalent of the conveyor belt (Kraft 1977; Greenbaum 1979). However, its prevalence at the time is questionable, because it was not even mentioned in some contemporary information systems textbooks (Burch, Strater and Grudnitski 1979; Davis 1974).

After 1980, three different research directions can be observed. First, within the mainstream of information systems development, signs of user dissatisfaction with existing systems led to the introduction of methods that supported user participation, such as prototyping (Boar 1984). As shown in section 2.3.4.6, however, prototyping can be interpreted in quite different ways.

Second, ideas which originated from computer science were transferred to information systems. The notion that specifications, as well as process and data models, could be described formally, and hence could be translated automatically to programs, led to the development of Computer Assisted System Engineering (CASE) tools and formal specification languages. Another trend was the evolution from object-oriented programming to object-oriented design. These subjects are discussed in detail in section 2.3.4.5.

Third, both researchers and practitioners discounted the central position of information systems development within the discipline of information systems. This is in line with De Wit's (1994) remark that SDM did not help in structuring the organization of automation. Consequently, researchers have called for a more thorough study of the relation between information systems and organizations. This subject is addressed in section 2.3.5. In practice, outsourcing of information systems development and the use of standard packages instead of special-purpose information systems have become increasingly popular. This theme is discussed in section 2.3.6.

2.3.3 Information systems development methods

The mainstream of the prescriptive information systems development literature asserts that the development of an information system should be considered as a project that must be executed according to a structured method. The minimum requirements for a structured method are:

a) The project is divided into a distinct number of phases or modules.
b) Each module is concluded by the delivery of a product, usually called a *milestone* or *deliverable*, most often in the form of a report or a program.
c) There is a schedule to determine when work on a module should be started.

Three advantages that can be attained by the use of a structured method to develop an information system are:

a) Lower cost and faster delivery.
b) Better quality. This may be more important than lower cost because an inadequate information system may not be used at all.
c) Lower risk, i.e. a lower standard deviation of expected cost, delivery time and quality of an information system.

The use of a structured method is justified either because it is beneficial in itself, or because it permits better management of human resources by a better distribution of work or by increased control of employees.

Structured methods are beneficial as such

The view that structural methods are beneficial as such is dominant in the computer science literature. For example, structured programming methods are recommended because they will benefit the individual programmer. According to this view, the main advantage of a structured method is that the system designer can exercise better control over his work. Advantages of better control that are

independent of a division of work are:

a) The risks of cost and/or delivery date overruns are limited because the project may be stopped or changed halfway.

b) Quality is maintained by control of quality in milestones. This improves quality because there is a higher probability that errors are detected and lowers costs because testing and repair costs increase more than proportionately with module size.

Structured methods permit division of work

When a system is developed by more than one person, some method must be used to divide the work. Accordingly, the prospect that division of work leads to shorter throughput times and more efficient use of specialized personnel can be considered a prime advantage of the use of structured methods. However, the use of a formally structured method is not a prerequisite for division of work because work can also be divided informally.

Structured methods allow increased control

In addition to the general advantages of improved control mentioned before, a structured method can also induce better results because employees will perform better if they know they are held responsible for delivering the right product at the right moment. This argument is based on the opinion that *direct control* is a better strategy for increasing productivity than *responsible autonomy*. Because workers prefer autonomy, the latter strategy will be followed when the labour supply is short (Friedman 1989), as it has been in information systems development for most of the past 30 years. On the other hand, managers may have a preference for direct control because of its proven success in industry. According to Kraft (1977), deskilling of the programming task, resulting in application of cheaper programming labour, was the chief motivation behind the popularization of structured programming in the USA. Greenbaum (1979), who voiced a similar viewpoint, added that the new

method did not provide any increase in productivity.

The appropriate research strategy to prove the value of structured methods depends on the justification of their use. If division of labour and increased control are the main arguments for the use of structured methods, research into the benefits and disadvantages of structured methods should be executed in the field, because those hinge on factors that cannot be replicated in the laboratory. On the other hand, laboratory research is appropriate if the benefits of structured methods are not dependent on the social effects of the workplace. Accordingly, in the subsequent sections it will be assumed that structured methods are beneficial as such.

2.3.4 Comparison of methods

2.3.4.1 Criteria for classification

Because there are hundreds of development methods, and new methods are continually proposed, there is also a large number of papers with advice on the choice of the best method. Moreover, many books and articles on new methods start with an introduction that explains why existing methods are unsatisfactory. This body of literature is of interest for our research because it helps to determine common features of methods that can be tested in laboratory experiments. To this end, we start with a classification of development methods according to five criteria:

a) The scope of the method.
b) The division into modules.
c) The sequencing strategy.
d) The mode of control.
e) The language(s) and tools used.

The number of possible methods is very large because a method is not simply defined by the choice of an option for each attribute. For example, a method can prescribe a different language and tool for each phase in the development. Our list of criteria can be compared to the seven basic elements of the Avison and

Fitzgerald (1988) framework, i.e *philosophy, model, techniques and tools, scope, outputs, practice,* and *product*. Languages and tools correspond to model, techniques and tools, The way a project is divided into modules is partly reflected in the outputs, and sequencing strategy and mode of control are part of the philosophy. Saarinen (1990) distinguishes *strategy*, *tools*, *formality of systems development* and *level of planning and management control*. Westrup (1993) lists three characteristics of information system development methodologies: *use of techniques*, *rigorous prescription of stages* and *managerial oversight*. Moreover, he points out that information systems development in practice may differ from the formal prescriptions in the method that is purportedly used. Avgerou and Cornford (1993) distinguish a focus on *professionalism and skill*, a focus on *procedures and sequence* and a focus on *tools and techniques*. The issue of professionalism and skill is not considered here because it cannot be adequately handled in a laboratory setting. Other comparative studies (Laagland 1983; Olle, Sol and Verrijn Stuart 1982) exclusively focus on the capabilities of languages and tools. The importance of the project management aspect of system development methods, i.e. division into modules, sequencing and control, was succinctly expressed by Brussaard in his comment on a draft proposal by Sølvberg (Olle, Sol and Verrijn Stuart 1982). Brinkkemper (1990) distinguishes *project management methods,* that describe what must be done, i.e. scope, division into modules, sequencing strategy and mode of control, and *development methods* which also describe how it must be done, i.e. the languages and tools used. Apparently, however, some authors limit the discussion of development methods to the latter subject.

The scope of a method delimits its application, either to a part of the life cycle of an information system, for instance information planning, or to a special class of information systems, such as decision support systems (Sprague and Carlson 1982). Because only methods with a common scope can be compared, we will only consider general methods pertaining to the full life cycle.

2.3.4.2 Division into modules

A project can be divided into modules by *division into phases*, *functional division* and *division into applications*.

Division into phases

Most development methods follow the well-known waterfall model, which divides a project into successive phases that become both more detailed and more formalized. For example, development proceeds from specifications in natural language by way of a graphical or tabular description of data and processes to programs in a programming language. In some methods, however, formalization by means of specification languages is prescribed from the very start of the project. In this way, successive transformations can be mathematically defined.

Functional division

Division into functional modules is less common than division into phases. Some early methods (Daniels and Yeates 1969; Stevens 1969) distinguished such separate modules as file design and input design. A model proposed in the MIS literature of the early seventies was the central database surrounded by processing modules. Usually, functional division is applied within a phase of a process that is primarily divided into phases. For example, many methods distinguish data structure design and process design within the design phase.

Division into applications

Division into application modules is not recognized as a systems development strategy because it is typically initiated by users. The pejorative term *islands of automation* associates this strategy with the dark ages of automation, but recently, it has become the norm in end-user-computing. It is also apparent in the manufacturing planning and control system (Vollmann, Berry and Whybark 1992).

2.3.4.3 Sequencing strategy

The sequencing strategy can vary only if a project is divided into phases, not if it is divided into functions or applications. There are three sequencing strategies, *linear strategy*, *iterative strategy* and *generic network strategy*, from which the first two are explicitly described in the literature.

Linear strategy

Development is conducted in a linear fashion if and only if:

a) Milestones are delivered in a predetermined order.

b) Once a milestone has been delivered, the preceding phase will never be repeated. However, repetition within a phase is allowed.

An example of a linear strategy is found in the school system. Although university teachers often advise their students to go back to grade school, the threat is never taken seriously. If, however, in a later stage failures from an earlier phase are detected, they are repaired at an increased cost, for example, when an MBA professor has to teach elementary arithmetic.

Iterative strategy

In the iterative strategy, phases are executed in a predetermined order, but a phase may be repeated after it has been finished. For example, the design phase may be repeated when the design causes insolvable programming problems.

Generic network strategy

In the generic network strategy, two or more phases may be executed in parallel. The phases are executed in any order that is compatible with precedence relations. Moreover, it is not necessary that all phases are executed. This strategy will be followed with a complex project structure.

2.3.4.4 Control mode

The control mode determines who decides on the quality of a milestone product. Three control modes are *control by management*, *control by peers*, and *control by users*.

Control by management

Usually, control is effected by someone who is directly responsible to management. This person is normally called the *project manager*. Olle et al. (1988) distinguish the roles of the *development coordinator*, who is responsible for the outputs of the process, and the *resource manager*, who attends to inputs. The project manager receives the deliverables and schedules the modules. Management may also delegate some control functions to actors in the system development process, for example the business analyst, or to other agents such as EDP auditors.

Control by peers

In a chain of materials, each customer controls the amount and quality of the products supplied to him. This strategy has been used within organizations as one of the instruments of internal control since the era of the medieval manor, where the thresher controlled the farmer, the miller controlled the thresher, the baker controlled the miller and so on (Chatfield 1977). In the system development process, control by peers implies that the systems designer controls the work of the specification writer, the programmer controls the work of the system designer, and, ultimately, the user controls the work of the implementor. Consequently, control by peers is best suited to a linear strategy in a phased development. A problem with this control method is that, in information systems development, a customer can evaluate only a restricted set of quality attributes, such as consistency and completeness. Accordingly, control by peers cannot be used as a substitute for, but only as a complement to management control.

Control by users

Each step in the development process can also be controlled by the users themselves. This may be advantageous if the project is highly uncertain. Because the users can change specifications at will, control by users will tend to coincide with an iterative strategy. This method is not appropriate for systems that are built for a large number of users with conflicting aims. Control by users also demands the application of languages the user can understand. The idea of providing such a language was already a motive for the design of COBOL (Sammet 1969), but to this day users are often asked to accept milestones containing unintelligible jargon.

2.3.4.5 Languages and tools

The language used to produce a milestone can be *formal* of *informal* and it can be *textual* or *graphical*. Informal graphics, i.e. pictures, are not widely used in information systems development. Formal texts include tables, programs in programming languages and mathematical formulas. Traditionally, informal texts have been used in the specification phase of the project, graphics, such as block diagrams, in the logical design phase, and programming and data base languages in the physical design phase. Proposals for integrating and automating the system development process have been proposed from the late sixties (Teichrow 1970; Bubenko, Langefors and Sølvberg 1971; Schneider and Wasserman 1982), but real progress in this direction was only made in the eighties. Examples are the introduction of data dictionaries, which are tables that are used in all phases of the project, the use of CASE tools, which mechanically produce graphic representations from formal texts, such as RIDL (De Troyer 1993), and the development of formal specification languages, such as Z (Spivey 1989), which purport to formalize the system development process throughout the system development life cycle. Though the advantages of formalization have been repeatedly expounded (Cohen 1989; Brinkkemper 1990; Hall 1990), they have not been generally accepted. The discrepancy between computer scientists and practitioners with regard to computer languages is highlighted by the fact that, in contrast to formal specification

languages, spreadsheet languages, such as Lotus 1-2-3 and Excel, are widely applied by end-users, but have received almost no attention from computer science research.

Another trend is the increasing use of *object-oriented* methods. At the time of the first conference on object-oriented programming systems, languages and applications, OOPSLA 86[7], the term object-oriented languages had been in use some time for a collection of languages that shared the notion of an *object*, which contains both data and the procedures to access that data. Subsequently, this concept was extended to include a larger part of the system development life cycle, including design (Booch 1986; Jacobson 1987) and requirements specification (Bailin 1989). Accordingly, object-oriented methods can be used throughout the system development process, which eliminates the need for a paradigm shift (Jordan, Smilan and Wilkinson 1994).

2.3.4.6 An example: prototyping

In the eighties, prototyping, which entails the use of a working prototype to clarify aspects of the information systems, particularly the user interface, has become a very popular feature in information systems development. Prototyping can be interpreted in different ways. One view, which is graphically shown by Royce (1970), is that prototyping embodies an iterative strategy because development progresses from system design to implementation of the prototype, succeeded by a return to the design phase and execution of the later phases of the production system. A second view is that prototyping adds a new development tool, the use of the prototyping environment. Third, prototyping can be seen as an enhancement of user control, because prototypes are discussed with the user rather than the project manager. Finally, prototyping can be seen as a new way to divide systems development into modules by separating user interface design from other design aspects.

---

[7] All proceedings of the OOPSLA conferences have been published as issues of SIGPLAN Notices.

2.3.5 Organization and information systems

According to the broad view on information systems outlined in section 2.1, the relation between organization and information systems is a prime concern of information systems research. Traditional theory holds that information systems development is determined by the shape of the organization, and that the structure of information systems does not influence operations. More recently, the idea of *strategic information systems* has been advanced, which proposes that organizations may change in nature because of the opportunities offered by new information technology. For example, in the late sixties, banks in the Netherlands have attracted large groups of new clients when lower data handling costs made this profitable. The notion of strategic information systems was brought to the attention of managers by McFarlan (1984), who generalized results from a number of cases. Since then, the growing interest in the subject has been shown by the creation of the *Journal of Strategic Information Systems*, as well as by the large number of articles on the topic in other publications. More recently, empirical research has been conducted by way of a survey (King and Sabherwal 1992) and a critical examination of case studies (Kettinger et al. 1994).

Within the organization, the buzzword has been *business process redesign* (BPR), which suggests that organizational changes may be introduced because of the potential of new technologies. For example, after the invention of the typing machine, the specialized typist entered the office, and after the introduction of PC's with word processors that even men could handle, typing jobs were quietly abolished. On the other hand, technological limitations can also force changes in organizations. For example, wholesalers have established self-service units to avoid expensive order processing. The history of published papers on BPR is strikingly similar to the publication record of strategic information systems. The concept was heralded to the business community by Hammer's 1990 article in the Harvard Business Review, and further explored in a large number of papers in the conceptual tradition. Apparently, the subject is still too new for the publication of results of empirical research. Recently, Davenport and Stoddard (1994) and Edwards and Peppard (1994) critically reviewed the basic assumptions of BPR, such as its novelty.

2.3.6 Outsourcing and standard packages

Recently, many companies have fully or partly outsourced information systems development, but the literature on outsourcing concentrates on the reasons for outsourcing, and does not specifically address outsourcing of information systems development (Arnett and Jones 1994; Elfring and Baven 1994; Grover, Cheon and Teng 1994). One view is that the quality of information systems obtained from outside suppliers depends on the correspondence between the delivered system and the specifications provided by the outsourcer. Another view it that the quality of an information system only depends on some innate characteristics.

A special form of outsourcing is the acquisition of standard packages instead of custom-designed information systems, which has seen a marked increase. Though there is an large number of papers on specific software packages[8], the subject has been neglected by the information systems research community. This is confirmed by Rands (1992), who notes the lack of comparative research on make-or-buy decisions and by Iivari (1990), who makes a similar remark with respect to implementation of application packages. Another topic that received some interest was methods for acquisition and implementation of software packages (Launi 1991; Tardy 1992).

There are two essentially different views on the implementation of a standard package. First, we may assume there is no essential difference between the regular system development life cycle and information systems development with a standard package. In both cases, development should start with specification and global design, and application of a standard package is considered a replacement for custom programming. Second, the standard package can be considered as a model that should be used from the very start of information systems development. According to the first view, implementation of a standard package will profit from clear preliminary specifications. According to the second view, the quality of an information system based on a standard package depends on the quality of the standard package and its correct application.

---

[8] On April 18, 1995, Excerpta Informatica listed 1295 articles and books on "software packages" that had been published in 1990 or later.

2.3.7 Laboratory experiments

Three factors are important in designing laboratory experiments in information systems development: the part of the system life cycle simulated in the experiment, the diversity of the simulated environment, and the number of different roles. The scope of the method should be covered by the part of the life cycle simulated in the experiment. The division of the project into modules and the sequencing strategy can only be tested in an experiment that covers a large part of the system life cycle and simulates a varied environment. For example, subjects can be given the task of designing and building an information system for a simulated company. In such an experiment, a choice must be made between a division into phases, a functional division and a division into applications, and if a division into phases is chosen, a linear or an iterative strategy must be selected.

Experiments with the mode of control require the same type of environment, and in addition, they require that the experiment involves different roles, such as users, managers, analysts and programmers.

Experiments with languages and tools can be executed with individual subjects within a single phase of the life cycle. For example, Harel and McLean (1985) compared the efficacy of different programming languages by an experiment that was restricted to the programming phase. However, research on this topic will certainly profit from embedding it in a larger context. For example, the results of the comparison of two programming languages may be quite different when they are used to build prototypes for users than when they serve to implement formally defined specifications.

Laboratory experiments with strategic information systems require an environment with competition between organizations, for instance a management game. The feasibility of laboratory experiments with business process redesign will be discussed in chapter 6.

Experiments with standard packages should cover the full systems development life cycle. Moreover, either the laboratory environment should be adapted to the commercial package chosen, or special packages should be built for use in the laboratory.

Laboratory research will be covered in more detail in chapters 3, 4, and 5. Chapter 3 discusses experimental methods in general, chapter 4 describes Infogame, the tool used in our experiments, and chapter 5 reports on results of actual experiments with Infogame.

2.4 Assumptions and Hypotheses

Because of its central position in the discipline of information systems development, we start our experiments with a test on the validity of the waterfall model. In the terms of section 2.3.4, we specify that information systems development is divided into phases, and that a linear strategy is applied. The experimenter ascertains that phases are completed in time, but does not evaluate the quality of the milestones. From the discussion in section 2.3, we derive two hypotheses that will be put to a test in chapter 5.

Hypothesis 1: In information systems development, the quality of a milestone product depends on the quality of the preceding milestone product.

Hypothesis 2: In an information system, the quality of a milestone product depends on its correspondence to the preceding milestone product.

Together, hypotheses 1 and 2 assert that application of the systems development life cycle is positively correlated to information systems quality. This does not necessarily imply that it is the cause of improved quality, because extraneous factors may induce both the use of the SDLC and an increase in information systems quality.

Our third and fourth hypothesis are derived from the discussion of outsourcing and standard packages in section 2.3.6. Hypothesis 3 pertains to information systems that are built to specifications by an outside contractor. Hypothesis 4 concerns the application of standard packages. In that case, specifications do not always exist, hence hypothesis 3 cannot be directly applied.

Hypothesis 3: User satisfaction with an information system is higher if there is a closer correspondence between specification and implementation.

Hypothesis 4: Users who prepare clear preliminary specifications will have more success in implementing and using a standard package than users who do not.

Finally, we use our experiments to additionally test the classical hypothesis from DSS effectiveness research (see 3.3.4):

Hypothesis 5: Organization success, as measured by company profit, is positively correlated with information system quality.

3 Experimentation and gaming

3.1 Introduction

Experimental research in information systems may be considered from the *experimental viewpoint* and from the *engineering viewpoint*. From the experimental perspective, which may also be called the *top-down* perspective, research should pertain to a given problem area, for example, the effect of colour in user interfaces, and it may use different tools, such as simple reaction tests and elaborate management games. From the engineering or *bottom-up* viewpoint, experiments are classified according to the tools used, for example, management games or role playing games, and with each class of tools experiments in different fields may be performed.

According to the experimental attitude, an experimenter should buy an existing tool on the market or order a new tool from an engineer. He is not interested in the use of this tool for other purposes. On the other hand, an engineer builds a tool, possibly because of the demand of an experimenter, and looks around for other uses of this tool. He is not interested in other tools to accomplish the same results. A typical conversation between an experimenter and an engineer is given in Fig. 3.1.

Experimenter:	I want to do an experiment on the importance of data accuracy in investment decisions. Can I use your management game for that purpose?
Engineer:	Yes, and you also can use my game for experiments on group interaction in investment decisions.
Experimenter:	That's not my field, but perhaps you can tell me whether I could use a simple manual game for my experiments.
Engineer:	Don't ask me, my specialty is the design of sophisticated management games.

Fig. 3.1: Conversation between experimenter and engineer

In the pure experimental view, research can never be hampered by lack of

suitable tools. If a tool is needed that doesn't exist, the experimenter will specify it, get the necessary funds and order it from an engineer. In my view, however, the value of a new research tool can only be assessed after it has been built and used. Accordingly, advanced research tools should be designed for a broadly defined problem area, and, if necessary, adapted for specific problems. For example, Infogame, the instrument used for research in this thesis, was originally designed to assess the value of decision support systems, but presently it is used to evaluate system development methods. This view has been commonplace in the natural sciences since the Renaissance, but not in information systems development, where the *computer laboratory* is often the exclusive domain of the computer scientist.

In section 3.2, experimentation will be treated from the engineering viewpoint, with an emphasis on gaming. Because all laboratory experiments with human subjects can be seen as games, the relative merits of gaming and other tools are not discussed. In section 3.3, experiments are treated from the experimental viewpoint, and in section 3.4, an attempt at a synthesis of the two views is made.

3.2 Gaming[9]

3.2.1 Definitions

Our discussion of games starts at a fairly high level of abstraction, because readers will intuitively understand the notion of a game, and will know many of the games that are used as examples in our classification. In sections 3.2.1 through 3.2.5, no reference is made to computers. These sections are necessary for the discussion of computer games in general in section 3.2.6, and management games in particular in section 3.2.7.

By this arrangement, our approach is different from the approach favoured in the discipline of *simulation-gaming*. By concentrating on equivalents of games, we

---

[9] A simplified version of section 3.2 has been published in Dutch as "Bedrijfsspelen" in Koorevaar,P., Oonincx, J.A.M., and P. Ribbers: *Handboek bestuurlijke informatiekunde* (Handbook of information systems), Samsom BedrijfsInformatie, Alphen aan den Rijn, 1995.

focus on competitive games with a single type of player, and neglect educational forms of unstructured play, such as role-playing games. In practice, games developed in the simulation-gaming tradition often involve more different roles than the games we consider, and they are also characterized by a greater reliance on unwritten rules and game administrator decisions. Moreover, many games from the simulation-gaming field pertain to relatively uncharted domains, such as the social and economic structure of the African village (Greenblat 1987), or family planning policies (Geurts 1981). This has two important implications. First, in simulation-gaming designs a large part of the effort is spent on preliminary research, whereas we assume that models can be based on textbook knowledge. Second, in the simulation-gaming tradition there is a fairly strict distinction between the use of a game in education to teach the basics of the field to the uninitiated, and use in research, where it is played with professionals in the domain. On the other hand, management games are usually played by students or managers with a general knowledge of the field, and hence they can be used for education and research at the same time. Moreover, simulation-gaming exercises often encourage a multidisciplinary approach (Duke 1982), whereas formal games are usually focused on a single discipline. Nevertheless, classification criteria described in the subsequent section, such as the level of abstraction, can also be used in simulation-gaming.

It is difficult to find a definition of the term *gaming* that is both theoretically sound and corresponds to its conventional meaning. If gaming is defined as an activity were participants can acquire benefits without directly contributing to welfare, the whole educational system may be called gaming because a valuable prize (a degree) is acquired by activities, such as doing examinations, that are intrinsically useless. On the other hand, according to this definition, a Soccer match between professional teams is no game, because the players' activities are essential for the sale of tickets, which are goods in the economic sense. Huizinga (1938) lists three characteristics of a game: it is played voluntarily, it is played within a limited span of time, and it is played according to fixed rules. As explicitly stated by Caillois (1958), the first condition excludes gaming by professional players. In addition, game playing by addicts should be excluded. The second condition separates gaming from such pastimes as gardening or embroidery, and the last

condition distinguishes games from unstructured play (Opie and Opie 1976). In this chapter, two aspects of the relation between games and society will be discussed. First, all games mirror some facets of society, which makes games a subject of research for historians and social scientists. For example, the rise of the general level of civilization in the nineteenth century can be inferred from the decline of games involving cruelty to animals, such as cat-beating or goose-pulling (Ter Gouw 1871; De Jager 1981)[10] [11].

Second, games offer training in skills that are currently not needed for practical purposes, but may be important for survival. For example, games and exercises have always played an important role in training for emergencies such as war or fire. Geurts (1993) emphasizes the use of games to help officials to adjust to the changing role of Government.

Though games have also been used to informally model human activities (Berne 1967), games as a model of social and economic systems are primarily studied in *game theory*, started by the work of Von Neumann and Morgenstern (1947). Game theory can be seen as a two-level abstraction. First the systems to be studied are represented by a game, then the game is represented by a mathematical model. In contrast, *gaming* involves participants who actually play a game. Although some games for which game theory offers a theoretical solution, notably the prisoners dilemma, are also used in experimental settings, the mainstreams of gaming and game theory have developed in opposite directions. Whereas game theory has gained in mathematical rigor and academic esteem, gaming has more and more found its supporters in the soft sciences and in practice. In view of the increasing mathematical flavour of information systems development research, it is surprising that only one paper on the application of game theory to this field could be found (Whang 1992).

The design of games for educational purposes has attracted much attention.

---

[10]In social history, these "games" are best known because of the notorious 1886 Amsterdam eel-heading riots, but the relevance of the suppression of eel-heading to those events is disputed.

[11]The question why Huizinga's *Homo ludens* neglected the games and plays related by Ter Gouw is beyond the scope of this thesis, but Huizinga took the same view on games when he inferred the loss of playfulness in modern times from the professionalization of Bridge.

Game reviews are found in the Gamesters' Handbook (Brandes and Phillips 1979), which lists both descriptions of simple games and the purposes to which that game can be used, and the Handbook of Business Games (Elgood 1981). However, there are few empirical studies on the effectiveness of gaming as an educational tool (Norris 1986; Geurts 1993).

3.2.2 A general classification of games

Games have been classified for different purposes. Thiers (1686) made a distinction between *jeux d' action* (sports), *jeux d' addresse* (games of skill) and *jeux de hasard* (games of chance) on moral grounds, because the last class of games was considered improper for christians in general and monastics in particular. He gave no unequivocal verdict on games that contained both skill and chance elements, such as Backgammon. The classification of Von Neumann and Morgenstern (1947) was intended to single out the games that would be the best candidates for mathematical analysis. In their definition, a game is always played by two or more persons, and it is either a sport game, where the player wins by his speed, strength and dexterity, or an inactive game, where a player wins by pure luck or by making the best choice from a set of possible moves. An inactive game where the player can make no choice at all or where he cannot make an intelligent choice, such as Roulette, is called a game of chance. All other inactive games, such as Chess, Bridge or Poker, are called games of strategy. The only difference between this classification and the one that preceded it by almost 300 years is that games with a chance as well as a skill element are definitely included in the games of strategy class. Recently, Van der Genugten and Borm (1991) devised a metric to determine the skill element in games with a chance factor because Dutch law permits those games that were forbidden to seventeenth century monks, but only in the hallowed halls of Holland Casinos (Van der Genugten 1993).

Klabbers (1987) suggested a taxonomy of games with the aim of improving communication between users and designers of different types of games. His classification is based on a large number of attributes, but it lacks precision because it makes no clear distinction between user aspects such as context of work and game

aspects such as rule base, and it combines lists of attributes and lists of attribute values. Moreover, it gives too much attention to ephemeral hardware and software attributes. Shubik (1975) presents no explicit classification, but he lists four attributes that can be used in a typology of game use, *purpose* of gaming, *disciplines* that use a game, and *formality of rules* and *richness of the environment* of the game itself. The latter two attributes will be discussed later on, but they are not used as primary dimensions here.

Our classification has two purposes. First, we want to provide some guidelines for game designers, especially designers of computer games, and second, the classification should help users to select appropriate games. Consequently, our classification is based on attributes that are important both from the viewpoint of the player and from the viewpoint of the designer.

The first attribute in our classification should distinguish sport games from inactive games. Two possible criteria are the qualities needed to win in the game, i.e. dexterity in sport games and intelligence in inactive games, and the role of time, because in most sport games the outcome depends on the relative speed of players, whereas in inactive games, the player cannot profit from speeding up a move. Because it is more relevant to computer games, we prefer the latter criterion, although it classifies such sport games as Golf and Snooker as inactive games. We call a game *time critical* if its result depends on the moment when a move is made, and *time free* if it does not. We assume, however, that in inactive games a move must always be made before a predetermined time limit. The analysis of inactive games rests on the assumption that the result of a play does not depend on the relative speed of the decision processes of players, which is implied by the literature on gaming (Von Neumann and Morgenstern 1947), but owes its explicit statement to the study of cooperating processes in operating systems (Dijkstra 1968). In the terminology of operating systems, a time free game is synchronized after each round, i.e. when all players have made a move, whereas synchronization in a time critical game occurs only sporadically, for example when a point has been scored. In a game where synchronization does not occur after every round, a player can improve his result by speeding up his moves, so such a game is time critical.

We extend the Von Neumann and Morgenstern classification to one-person

games by calling a one-person sport game a *sport performance*, a one-person game of chance a *lottery* and a one-person game of strategy a *puzzle*. A *sport contest* is a set of sport performances where the best player is determined by direct comparison or by measuring the results of all competitors against a common yardstick such as time spent. Connoisseurs will see the distinction between a cycle race, which should be classified as a sport game, and a skating race, which is a sport contest. A lottery contest is equivalent to a simple lottery. A puzzle contest is called a *quiz*. Contests where participants do not engage in single-person games, and the result is based on the subjective opinions of jury members are called *artistic competitions* or *beauty contests*. The fact that some artistic competitions, such as figure skating or dressage, are traditionally labelled as sports should not confuse us. This classification is summarized in Table 3.1

Table 3.1: Types of games

		Game	Contest	Performance
Time critical		Sport game	Sport contest	Sport performance
Time free	Chance	Game of chance	Lottery	
	Strategy	Game of strategy	Quiz	Puzzle

3.2.3 Additional classification criteria

A further classification of each group of games can be based on four attributes that are important in the design of computer games, but can also be discerned in traditional games, *playing sequence*, *information*, *randomness* and *abstraction level*. All attributes can be used in the classification of games of strategy. In other games, an attribute may have only one value. For example, a game of chance without random factor is no game at all. A similar classification for puzzles will be given in section 3.2.5.

Playing sequence

In time critical games any playing sequence is allowed during the course of the game, and the actual playing sequence is determined by the players. Time free games can be divided into sequential games, where players have to make their moves in a fixed sequence, and games played in rounds, where the result is computed after all players have made their move. In a game played in rounds, the result should not depend on the order of the moves of the players, because otherwise the game would be time critical. Moreover, moves must be hidden from other players during a round, because a player can make better decisions if more information is available. Accordingly, the player who enters his decisions last would get the best results, and this would prolong the game infinitely unless some premium is set on quick action, which in turn would make the game time critical.

Playing in rounds is faster than sequential playing, but it is difficult to design a board game where a move is hidden from opponents. In card games this can be effected by putting cards face down on the table. In computer programs playing in rounds is easily implemented, and consequently it is often used, for example in management games. Games of chance can also be played sequentially or in rounds. In a game of chance played in rounds, such as Roulette, there is no need to conceal a move during a round, because opponents cannot profit from knowing it.

Information and randomness

Two related classifications are the distinction between games where each player has complete information on the state of the game and games where he has not, and the distinction between games that contain a random factor and games that do not. In time critical games, a random factor is automatically provided because the exact timing of a move cannot be determined, and it is not important whether a player has complete information on the state of the game because the time available for decision making is necessarily too short to use all available information.

A game of chance without a random factor is no game at all and whether players have complete information on the state of a game of chance is important only

from a psychological point of view. Accordingly, the discussion of the role of information and randomness will be confined to games of strategy.

A random factor may be determined when it is needed, for example by throwing a dice, or it may be determined beforehand, for example by preparing a stack from which the upper card will be drawn. Neither the result of a throw nor the result of a draw gives any clue to the sequence of future throws or draws. Cards and dice differ because the number of cards is strictly limited. Some games explicitly aim to exhaust the stack, in other games any card is put back at the bottom of the stack, which makes the stack of cards behave like a pseudorandom number generator with a cycle length equal to the number of cards.

A random factor is equivalent to an additional player without a known strategy. Whereas the moves of a regular player necessarily leak some information to his opponents, the moves of the additional player do not reveal anything about his behaviour in the future. Normally, the additional player has nothing at stake either. If he has, like the owner of a Casino, sufficient checks should be built in to prevent him from translating his interest into a playing strategy. The notion of an additional player as a source of random moves can be noticed in ancient religions that saw the hands of the gods in dice. Of course, if a supernatural being decides on the number thrown or the card drawn, that being is thought to have a stake as well as a strategy, so players try to divine its intentions and to influence its acts. Though in what we call the civilized world such beliefs have been extinct for thousand years or more, they still persist in the form of magic and superstition. However, according to Janssen (1985), there is no support for the alleged descent of the Tarot card from ancient religions. The classification based on information and randomness is illustrated with examples of games of strategy in Table 3.2.

Table 3.2: Information and randomness in games

	No random factor	Random factor
Complete information	Chess	Monopoly
Incomplete information	Stratego	Bridge

Stratego, invented by Mogendorf when in hiding during the German occupation of the Netherlands (Verhulst 1985), employs a special type of piece, which unlike cards in the hands of a player, can occupy a fixed position, and unlike open cards or Chess pieces, has a signature that can be seen only by its owner. Each player starts by distributing his pieces over his side of the board. If a player moves a piece on a field occupied by an opposing piece, the ranks of the pieces are called out and the lower-ranking has to leave the board. The higher ranking piece remains on the board, but its identity is henceforth known to the opponent. A player wins if he captures a special piece, the flag, of his opponent.

Among aristocrats and intellectuals, games like Chess have always commanded the highest esteem. On the other hand, games like Monopoly are popular with the working class because its view of the world is better reflected in a game with a random factor. The popularity of Bridge may be attributed to the role it allows for subtle modes of communication. The scarcity of Stratego-like games is difficult to explain.

It should be noted that the random factor that characterizes Bridge as a game is removed from Bridge tournaments by a device that is known in the simulation literature as common random numbers (Kleijnen 1988). This entails that tournament results are determined by comparing the results of players with the same hand. Accordingly, a Bridge tournament can be considered a gaming contest. This does not, however, change the character of the Bridge game itself, because the same device can be applied to all games with a random factor. For example, in a Monopoly tournament, a games master could throw the dice for all players in a hall, and results would be determined by comparing the results of players who apply the same throws.

Symmetric and asymmetric games

In symmetric games, all players essentially follow the same rules, whereas in asymmetric games, players play different roles. Most popular games are symmetric, one of the few examples of an asymmetric game is Fox and geese (Grunfeld et al. 1975). It is difficult to design an asymmetric game with even

chances for all players. Because equal opportunities are very important in gaming, players in asymmetric games, such as Tennis, are usually assigned alternate turns. The principle of even chances hampers the design of games that realistically depict society. For example, in management games, competing firms normally start with the same resources, which is not an accurate image of the real world.

Games of chance may be superficially asymmetric, for example in racetrack betting, but if the expected return is equal for all parties, such a game can be considered a symmetric game on a higher level.

Level of abstraction

All games simulate serious activities in some way, but there is a wide range in the level of abstraction. We distinguish the *apparent* level of abstraction, which is determined by the similarity of the playing materials to real life objects, and the *intrinsic* level of abstraction, which is determined by the influence of real-world phenomena on the course of the game. The notions of apparent and intrinsic level of abstraction can be demonstrated by a comparison of Monopoly and Chess. On the apparent level, Monopoly has a fairly low level of abstraction because of its association with real-life objects such as money, streets and houses, whereas Chess has a high level of abstraction because of its symbolic pieces. However, the intrinsic level of abstraction of both games is just as high, because the rules of Monopoly are totally independent from real-world phenomena such as the current prices of houses, which is underscored by the intrinsic equality of all local versions. The real differences between Monopoly playing and Chess playing stem from the effect of the random factor and the large set of rules in Monopoly.

The importance of the apparent level of abstraction from a psychological point of view is demonstrated by the old Dutch game of Boerenschroom, a very tedious card game that may get some interest from players because of the pungent social comment implied by the fact that the lawyer usually ends up winning all the spoils (Botermans, Visser and Dekker 1991). Other games which derive their popularity at least in part from their low apparent level of abstraction are Games of Goose, which have been designed to stimulate education and informal discussion on

a wide range of subjects (Vogel 1981).

There are two classes of games with a low intrinsic level of abstraction. In word games such as Scrabble, success depends on knowledge of words in a real language, and the most recent dictionary in that language may be considered a supplement to the game rules[12]. In quiz games, that date back at least to around 1960, when "Go to the head of the class" was published, but really caught on with Trivial Pursuit, players win on account of their factual knowledge. Unlike word games, quiz games do not adapt to a changing world, and consequently, old versions of Trivial Pursuit change into games with a higher intrinsic level of abstraction, whereas old Scrabble boards have no trouble with new words. As the term implies, quiz games are not real games, but knowledge contests, because players cannot influence the chances of competitors.

Time critical games usually have a high level of abstraction, because they are modelled after activities that nowadays are automated. This is also shown by the fact that professional sport players receive special training that serves few purposes outside the arena.

The intrinsic level of abstraction of games of chance is necessarily high, because any game of chance is equivalent to drawing a random number from an appropriately defined set. Games of chance with a low apparent level of abstraction, such as football pools, lure the player into believing he is playing a game of strategy.

3.2.4 Evaluation and rules

Rules in games differ from rules in real life. Games have a finite number of unambiguous rules that are observed by all players. A player who disregards a rule is said to cheat and is punished, usually by being locked out of the game. In games with incomplete information, players can cheat by obtaining the missing data in a devious way, and in games with a random factor, players can try to guess or

[12] It should be noted that a version for children where players must fill in printed words has a higher intrinsic level of abstraction.

influence that factor by such devices as marked cards or loaded dice. Cheating is especially profitable in pure games of chance, and consequently, Casinos employ the equivalent of a police force to prevent it.

In real life, formal rules are supplemented by a fuzzy set of unwritten rules. Anyone who disobeys the written rules may be legally punished. Someone who disobeys the unwritten rules is guilty of unfairness. In most societies only the most brutal forms of unfairness are legally punished, though it is normally disapproved by the public. Unpunished unfairness is possible because rules may be interpreted in more than one way, which is more likely when the set of rules is large and rules are formulated in vague language.

In games, unfairness is possible if and only if the rules are not clearly defined. This occurs in sport games, and in games with a low intrinsic level of abstraction, where the written rules of the game are supplemented by informal rules derived from real world customs. For example, a Scrabble player may assume that indecent words are forbidden whereas his opponent has no such scruples. Unfair play is also possible in games with a very large set of rules. In such games, players who start without full knowledge of the rules may easily be deceived by more experienced or cunning players. In the majority of games, however, there is a sharp dividing line between cheating, which is always punished, and playing according to the rules, which is considered fair. This is a salient characteristic of gaming and therefore arbiters, who must ensure the strict observation of formal rules, have a prominent role in games.

A game is evaluated by assigning an ordinal number to each player. In a zero sum game, the sum of the results of all players is zero. A special case of a zero sum game is a two-person game where the possible results are $\{-1,0,1\}$, i.e. loss, tie or win. The result of a game can be decided when a special ending state has been reached, such as checkmate in Chess, or it can be determined from the state of the game after it has been decided independently that the game has been ended, as in Go, Reversi or Bridge. There are also games, such as Scrabble, where an additional accounting scheme is needed to determine the result.

The proceeds from a game can either be paid to the winner after each game, or the result of a game can be used as a score in a series of games, sometimes called

a match or a tournament, and only the total result of that series is paid in real-world currency. In sport games, elaborate payment schemes have been devised. For example, in the *Tour de France,* the main European bicycling event, prizes are allotted for such functions as the number of days a participant has been the current frontrunner, and prominent cyclists get the opportunity to participate in lucrative races in the remainder of the season. Such payment schemes encourage a strategy which does not purely aim at winning each game.

3.2.5 Puzzles

Not all criteria used in the classification of games can be used to classify puzzles. Playing sequence is irrelevant because there is only one player. Formally, a puzzle with complete information is no puzzle at all, but information about the state of a program can be considered incomplete if the player cannot derive a winning strategy from it. Because there is no adversary, there is no difference between information that is unknown to a player but is known to one of his opponents and a random factor that is completely unknown.

Puzzles can be classified into deterministic puzzles, where the state is always fully known, and stochastic puzzles, where the state is only partially known. In a deterministic puzzle, such as peg solitary, a player can recognize a state from which he knows a path to the winning state, so there is no challenge in solving such a puzzle twice in the same way. In a stochastic puzzle, such as a solitary card game, the player is never sure whether a state is identical to the state from which he found his way to the winning state. A stochastic puzzle can also be considered as a set of puzzles where the player does not know which one he is trying to solve.

Any puzzle is equivalent to a two-person game with an imaginary opponent. In a deterministic puzzle the strategy of this opponent is known to the player. For example, in a Chess puzzle, it is assumed that the opponent always makes the best move. In a stochastic puzzle, the strategy of the imaginary opponent is unknown.

Like games, puzzles can be classified according to their apparent and intrinsic level of abstraction. Puzzles with a low intrinsic level of abstraction, such as crossword puzzles, especially cryptograms, can only be solved with a knowledge

of the environment, whereas puzzles with a high level of abstraction can be solved by any intelligent being. Accordingly, puzzles used in intelligence testing should have a high level of abstraction to avoid a cultural bias, but for puzzles used to test suitability for employment in a well-defined environment, such as the diplomatic service or the sales organization of a computer firm, a low level of abstraction may be preferred.

3.2.6 Computers and games

3.2.6.1 Classification of computer games

A computer can be used in games as a playing tool or as an active participant. In a one-person game we assume that the computer acts as the imaginary opponent. In a multi-person game we consider the computer as a playing tool because a program for a multi-person game with the computer as an active participant can be decomposed into a program that uses the computer as a playing tool and a program that simulates an active player.

By crossclassifying the two classes with the classification into time free and time critical games and a classification according to the level of abstraction, we define the computer applications to games in Table 3.3.

Table 3.3: Classification of computer games.

	One player		Two or more players	
	Time critical	Time free	Time critical	Time free
Equivalent game	Sport performance	Puzzle	Sport game	Game of strategy
High level of abstraction	Arcade game	Game playing machine.	Computer race	Computer game board
Low level of abstraction	Simulator	Adventure or Simulation game	Real-time management game	Management game

In our discussion, simulators and computer races will be treated with arcade games, game playing machines will be combined with computer game boards, and real-time management games will be described in the section on management games.

The relation between gaming and *simulation*, a technique that supplies numerical solutions to models (Kleijnen and Van Groenendaal 1992), is complex. First, gaming may use simulation to create a more natural environment for players. For example, a management game usually includes a simulated market. Second, simulation of the technical aspects of an information system may parallel gaming to study human aspects. For example, a simulation model of a database may be used to calculate system response time, while a game may determine human response time. Third, in many simulations, human reactions must be modelled, and gaming may help in developing a more realistic model for those reactions.

To add confusion, the term *simulation* is also used for unstructured games that defy computerization. Its use will be discussed in section 3.2.7 under the more specific label *social simulation*.

3.2.6.2 Computer puzzles

Computer programs can either represent a puzzle or actively solve a puzzle. In any case we will have to restrict the field of computer puzzles to puzzles with a high level of abstraction, otherwise this relatively unimportant branch of recreational informatics would include the whole art and science of computer programming, because data structures can be considered as representations of puzzles in computer memory and algorithms as procedures to solve puzzles.

Puzzles may be solved by computers in three ways: by brute force, i.e. by trying all possibilities, by a simple algorithm that straightforwardly finds the solution and by a complicated algorithm or a heuristic that looks for the solution in a clever way. Examples of all three methods can be found in numerous textbooks on programming methods.

3.2.6.3 Game playing machines and game boards

A program that uses the computer to represent a game board must accept the moves of players, check whether a move is legal, determine whether the game is ended, and if so, compute the result, and represent the state of play. For games with complete information, the state can be represented on a screen that is visible to all players. For games with incomplete information, a private screen should be provided for each player.

Because in most popular games playing with traditional materials is attractive, computer game boards do not improve many games, except for absolute beginners. However, some games, such as Reversi, that did not gain the popularity they deserve because they are difficult to handle manually, should profit from computerization. Some other applications of computer game boards are the use of a program to check the moves in Chess matches before they are stored in a database, and the use of a computer in official Bridge tournaments to suppress illegal leaking of information by movements, voice inflection or timing.

In two ways, computer game boards offer scope for new types of games. First, more complex games may be developed. However, there is hardly a demand for more complex games with a high level of abstraction. Second, the computer may control the information supplied to the participants in a way that cannot easily be implemented in other ways. For example, a computer version of Monopoly could ask for payment without telling the player whom he should pay.

Game playing machines have the longest tradition in the field of computer gaming. Almost from the inception of the first computer, the suggestion has been made that a computer could play Chess and since then a debate has raged on the question whether or when a computer Chess program would beat the human Chess champion. The early history of computer Chess is recapitulated by Van den Herik (1983). Starting with a computer game board it is easy to design a program that plays a two person game by recursively trying all moves for the computer and its opponent and choosing the first of a winning or nonlosing sequence of movements. It is also easy to prove that, in a nontrivial game, execution of such a program on the most powerful computer would take longer than the expected life of the universe. A

satisfactory program must evaluate the intermediate states of the game and choose the move that will lead to the best state. The design of game-playing machines is usually considered part of artificial intelligence. Apart from matches between computers and human players, matches between computers have been organized, such as the annual computer Chess tournaments. Competition between game-playing programs is especially attractive for games with a random factor because its influence can be excluded by a match of 100 or 1000 games, which would be too boring for human players.[13]

3.2.6.4 Simulators and arcade games

Technically, there is no difference between simulators, with flight simulators as the best known example, and arcade games, and in practice some simulators, such as the popular flight simulator for the IBM-PC, are used for amusement rather than for serious educational purposes. Computer arcade games have been developed from mechanical arcade games or slot machines. The first generation of arcade games was played on specially developed machines in playing arcades and other public places (DeFanti 1984). Because of the increase in computing power, PC's can now execute very similar games. Arcade games are time critical games where the speed and accuracy of the player determines his success. Some games use a keyboard for input whereas other games are designed for pseudoanalog input devices such as game paddles, joysticks or mice. We call such devices pseudoanalog because their input is converted to a digital value before it is used in computation. For example, the input from game paddles is often converted into a value in the range 0..255.

The design of arcade games has developed into an art which may use the newest technology but has not yet developed a scientific base of its own. Any arcade game must comply with the following rules:

---

[13] A computer game board with four simple strategies was published in Casimir, R.J., "Mens-erger-je-niet", *Personal Computer Magazine*, Vol 4 No 3 (March 1986), p 40-41,95-96.

1) The result of the game must depend on the moment an input signal is supplied as well as on the input signal itself.

2) The program must be correct irrespective of the timing of the input. As a consequence it should also be correct if the player supplies no input at all.

An arcade game can be conceptualized as a system of two interacting processes, one process that continually changes the state of the game, and one process that receives input from the player. It is clear that mechanisms based on the assumption of indefinite speeds, such as semaphores (Brinch Hansen 1973; Ben-Ari 1982), cannot be used in arcade games. However, all interactions in arcade games can be defined by references to shared memory. This can be proven informally as follows:

The state changing program executes a sequence of statements, and each statement may respond to an input from the player. If a player produces an input during the execution of a statement of the state changing program, the input program deposits a value in a memory location and the next instruction of the state changing program will consult that location, so there is no need for any other mechanism to establish interaction between the two processes. In the operating systems literature, this is known as busy waiting (Brinch Hansen 1973; Ben-Ari 1982). However, whereas the state of a process in an operating system is frozen when it awaits a signal and consequently the processor time spent in waiting is unproductive, the state describing the background in an arcade game continually changes while awaiting an input, so the processor time used during waiting is not really lost.

Verification of arcade game programs decidedly differs from verification of programs that agree to a formal specification. An arcade game program can and should be proven to stay within a given state space and, especially if played on a coin-operated machine, it should be proven to terminate within a given time as well, but in other ways it cannot be proven to conform to specifications that users deem important.

From the user's point of view, two characteristics of arcade game programs are of paramount importance, the quality of the graphic representation of output and the speed of the state changing program. The quality of the graphic representation

cannot be formally specified. We may, however, expect that users will demand the highest quality available. The speed of a game should be finely tuned. A game which is too fast degrades into a pure game of chance. A game which is too slow loses its challenge for the user, or at best, changes into a time free game.

The speed of a game is determined by the number of statements executed for a state change and the time needed for the execution of a single statement. Consequently, it increases with processor speed, and it is decreased by the use of higher level languages, especially interpreted languages, complex operating systems and inefficient algorithms, as well as by the inclusion of states that contain more information, for example detailed backgrounds. Accordingly, to prove any assertion on the speed of an arcade game, besides the game program itself all layers between the game program and the physical machine should be studied. Because constant speed in a game is just as important as the average speed, special attention should be given to anomalous behaviour, such as long delays for garbage collection at unpredictable moments. The warning that time-sharing systems are unsuitable for arcade games is still true, but in the era of the ubiquitous PC it is as irrelevant as a discussion on the relative merits of the penny-farthing and the safety bicycle.

3.2.6.5 Adventure and simulation games

In adventure and simulation games, a computer program simulates an environment that is influenced by the decisions of the player. To this end snapshots of the environment are presented to the player. In general, adventure games are designed for amusement and model a fantastic world whereas simulation games are designed for research or education and model some part of the real world. Accordingly, adventure games have a higher level of abstraction than simulation games. Adventure and simulation games differ from arcade games because the state of the play is frozen while the player takes his decision and communicates it to the computer. As a consequence, the preoccupation with speed that is characteristic both for the designer and the player of arcade games is absent from adventure and simulation games. In general the time used to compute the state where the player will have to make a new decision will be short in relation to the time needed by the

player to make that decision. Consequently, the first adventure games could be developed for time sharing systems.

The time free character of adventure and simulation games implies there is no fixed relation between real time and simulated time. A player in an adventure game may quietly spend a minute or more pondering whether to shoot the tiger or to run away, but he may also order the construction of a large building and find it completed after a few seconds. Game designers or players who do not like this abstraction from real time should turn to arcade games.

From a formal point of view, any simulation or adventure game can be described as a finite state machine where the player supplies the input signals after being provided with information on the state of the machine. With this model, which also encompasses scenario's, expert systems and computer assisted learning systems, some elements of style in simulation and adventure games can be traced.

With regard to the inputs, we can distinguish games with simple inputs that move in small steps and games with complicated inputs that move in large steps. Step size and input complexity correspond because the extent of the changes in the simulated world determines the time a player is ready to devote to a decision, and this time in turn is related to the length of the input sequence he can produce. In a long input sequence, any element can be corrected before the final signal is given, and partial decisions can be made in any order. If the user interface of a game requires correct inputs in a fixed sequence, a player will not supply any input before it is completely specified, so lack of correction facilities in the input program is only a nuisance.

Because the user interface itself can be described as a finite state machine, any machine with complex inputs is equivalent to some machine with simple inputs and a number of additional intermediate states. Consequently, any game with large steps is equivalent to a game with small steps where only some steps produce output. In general, adventure games are played in small steps, whereas simulation games, especially games modelling social systems, are played in large steps. This reflects the complexity of decisions in social systems, where, to quote a simple example, a new employee cannot be hired without determining his pay, his tasks, his place of work and various other details. Moreover, simulation games are often played by

teams, and games that move in large steps are more suitable for team playing.

Syntactically, adventure games often allow the input of sentences in some form of natural language and may even leave the player uncertain about the precise syntax and semantics of that language. On the other hand, simulation games usually ask the player for specific inputs and correct erroneous entries. The state of the game can be represented by pictures, text or numbers. In general, adventure games use pictures and natural language text whereas simulation games more often use numbers. However, with regard to input and output, the difference between adventure games and simulation games may disappear because of the standardization of input and output forced by graphical user interfaces. Both adventure and simulation games may display a small or large part of the information present in the state. Of course, a game which displays all information in the state, including the sequence of random numbers that will be drawn in future states, is equivalent to a deterministic puzzle.

A finite state machine can either be described by a full description of all possible states and an enumeration of all cells of its transition table or by a program defining the rules according to which the states and transition tables can be constructed. In practice, the program for any game will be based on a mixture of the two methods, the first method being prevalent in adventure games, the second in simulation games.

A finite state machine that delivers a result by reaching a special ending state is called a finite automaton. Adventure games are often modelled after finite automata because they set the player a special task, such as the quest for a treasure. Simulation games more often may be stopped in any state and define the result by evaluating that state.

Simulation games have been extensively used in information systems research. For example, the Minnesota experiments (Dickson, Senn and Chervany 1977) were based on five simulation games. There are also ample possibilities to exploit adventure games for research, but no such research is known to the author.

3.2.7 Management games

Management games caught the attention of management and academics in the late fifties with the introduction of a business game by the American management Association. Some enduring characteristics of this game are that it is played in rounds, that company profits stem from sales that are determined by the division of a known total demand over competing products, and that there is no explicit criterion to select the winner of the game. In the remainder of this thesis, a game like the AMA game is called a *conventional management game*. The description of the AMA game (Ricciardi et al. 1957) depicts a bright future for the management game which is even compared to the airplane. The fact, however, that the text is still remarkably modern shows that, apart from the formidable drop in computer prices, management gaming has made no great strides. This is confirmed by Burgess (1994), who identifies *publication*, *proliferation* and *downloading to PC's* as the main events in the development of management gaming since 1962. He also mentions that many users still design and program their own management games, which can be seen as a sign of the immaturity of the field. The trend in management gaming is also shown by the development of the management game MAGEUR, which was first designed for a stand-alone minicomputer[14], and subsequently was adapted to a time-sharing minicomputer, a stand-alone PC and a PC network[15]. Though both the user interface and the internal structure of the game program have been changed several times, there have been no essential changes in the underlying model.

In the fifties and early sixties, the high cost and limited availability of computers acted as a deterrent to the use of computerized games and consequently some games, for example the McKinsey Business Management Game (Andlinger 1958), were designed as manual games. Meanwhile, the AMA game was the basis

---

[14] Casimir,R.J., *Ervaringen met een bedrijfsspel* (Experience with a business game), Unpublished Masters Thesis, Netherlands School of Economics, Rotterdam, 1970.

[15] RES, a simple version of MAGEUR, is described in "Bedrijfsspelen", in Koorevaar,P., Oonincx,J.A.M. and P. Ribbers (eds): *Handboek Bestuurlijke Informatiekunde* (Handbook Information Systems), Samsom Bedrijfsinformatie, Alphen a/d Rijn, 1995, C2630:1-46.

for more complex games, such as the Carnegie Tech Management Game (Cohen et al. 1962). A survey of the development of management gaming in the late fifties is given by Cohen and Rhenman (1960). Their paper also contains a section on the history of war gaming, which mainly follows an earlier article by Thomas (1957).

Instead of following the usual procedure of copying or extracting these papers, we found it quite rewarding to study the original literature[16]. Von Aretin (1830) prefaces his detailed description of a new war game with a concise review of war games from 1664 to 1825, which showed a decreasing level of abstraction and an increasing complexity. His assertion that games are a better means of instruction than "listening to long, tiring, half understood and soon forgotten lectures or hour-long browsing through books" could be copied in any present-day game manual. He also stressed the importance of the relation between the level of abstraction in a game and the purpose of gaming. For example, he thought Von Reiswisz's game contained so much detail that it only could be used for training on the tactical level.

Meckel (1873) also lists many advantages of war games over other means of instruction, such as the opportunity for exercises in giving and taking orders, which agrees with the modern view that management games should be primarily used to stimulate team work. He preferred decisions made by arbiters over decisions by game rules or dice throws, because he thought the latter led to unrealistic results, and consequently advocated what is now called *free-form gaming* (Shubik 1975).

In comparison to other games, the principal characteristic of management games is their low intrinsic level of abstraction. Most of the differences between management games and other games with a comparable apparent level of abstraction, such as Monopoly, are derived from this characteristic.

First, management games are played to learn something about the world that is modelled in the play whereas other games are played purely for entertainment. Consequently, participants in a management game can use their knowledge of the outside world when playing the game and can use the knowledge gained in the game in the outside world. Because participants tend to behave in the game just as they would in the outside world, management games can be used for research into

[16]The books of Von Aretin and Meckel were examined at Göttingen University library.

manager behaviour. A game can teach on widely varying levels of abstraction. On a high level of abstraction, it can teach business students to carefully study the use of sales instruments, such as product quality, price, and promotion, and on a low level of abstraction, it can proclaim facts or prejudices such as the notion that detergents should be advertised on daytime television. The intrinsic level of abstraction does not necessarily correspond to the apparent level. For example, the Carnegie Tech Game is modelled after the packaged detergent industry, but players are warned not to use their knowledge of this industry, and a quaint mismatch was found in an unnamed game that called its product "personal computers" but allowed for a yearly price increase equal to the rate of inflation.

A corollary of the educational aim of management games is that the designer of a management game should prevent that it is played for its own sake. Attention for successful play is an indication of the status of a game with a higher level of abstraction, as shown by the vast literature on Chess (Kruijswijk 1974), but advice on how to play a specific management game is detrimental to its use as an educational or research instrument. The fact that too much popularity can destroy a management game, or at best change it into a general game, was overlooked by Courtney, DeSanctis and Kasper (1983) in their case for a common gaming simulator. The evolution in the realm of crossword puzzles, where the appearance of special dictionaries has made solving a crossword puzzle a less sure test of proficiency in a language, can be seen as a warning sign. A game designer can prevent a change in that direction by providing a large number of parameters that can be modified by the game administrator.

Second, there is no obvious way of ranking results in management games. In this respect they resemble the real world, where a large number of variables can be used to measure the success of a firm, such as profits, market share, or solvability. Teach (1990) called the use of profits as a yardstick of success a *false prophet* because it tends to foster tactics to increase short-term profits at the expense of long-term goals. However, his proposal to use forecasting ability as a measure equally distorts the game by encouraging risk-avoiding tactics. On the other hand, the problems named by Teach are largely overcome if cumulative profits are replaced by the discounted value of expected profits, which can be derived from

such data as past profits, product quality, market share and quality of plant and labour force. Profit as a yardstick was also criticized by Estes (1986). In his opinion, business games may teach students that ruthless exploitation of workers does no harm to a firm. However, we may assume that game rules include minimum wage standards that are more rigidly enforced than in most Western countries.

Third, management games can profit from the use of complicated models to increase the similarity of the game to reality and discourage players to start playing the game for its own sake. Thus, unlike game boards for traditional games, management games can greatly profit from computerization. Moreover, because management games are not adapted from games played with traditional tools, but are usually developed specially for a computer, they can use a representation that is suitable for the computer. Normally, the output given to players resembles the output of information systems in comparable real-life situations.

The use of management games in information systems education was already proposed by Courtney et al. (1978). In an experiment, students had to design an information system for an existing management game. Later, a query facility was added to the same management game, BML (Courtney, DeSanctis and Kasper 1983). Similarly, the MAGNUS management game was integrated with a Participants Support System (Yeo and Nah 1992). Whereas the design of simulation games is so simple that they can be specially designed for experiments, most management games used in information systems research experiments are designed for other purposes. The design of Infogame, which will be described in chapter 4, was motivated by the view that the use of conventional management games would hamper research in information systems because they deliberately condense information.

Conventional management games or *standard composite business games* (Burgess 1994) are based on a rather simple model of the world, and for that reason, Van der Meer and Roodink (1991) argue for the use of *social simulations* instead. Social simulation is usually considered in the context of simulation-gaming, but a social simulation can also be regarded as a management game that differs from a conventional management game by the number and role of players, the underlying model and the evaluation of results. For example, social simulations often foster cooperation rather than competition between teams. Moreover, social simulations

may use flexible rules interpreted by arbiters instead of rigid rules enforced by a computer program, and they may use traditional playing materials instead of computers. Because all these characteristics are mutually independent, there is, in our view, no sharp dividing line between social simulations and management games.

3.3 Experimental research

3.3.1 Introduction

Experimental research in information systems has been concentrated in a small number of fields, from which two fields will be reviewed in more detail. First, attention is given to the study of human-computer interfaces. There is a large body of experimental research on the effectiveness of tables versus graphs, on the advantages of different types of graphs, on the use of colour, etc.. The overwhelming part of the published literature in this field consists of experimental research or reviews of experimental research. Two subjects have been chosen as examples. *Reading text from screens* is discussed in section 3.3.2 and *the use of graphs versus tables* in section 3.3.3. In section 3.3.4, experimental research on the effectiveness of Decision Support Systems, including Group Decision Support Systems, will be discussed. In this field, only a small part of all research is done in the laboratory, because it is also addressed by a large body of case and field research as well as theorem proof research in the management science tradition.

3.3.2 Reading text from computer screens

The field known as *reading text from computer screens* encompasses comparing reading texts from screens and from paper and comparing different presentation attributes, such as letter type or colour. It has been the subject of an essentially descriptive survey by Weldon and Mills (1987), and a more critical review by Dillon (1992). Both are based on published papers in specialized journals and we assume they give a fair view of the field. As the Mills and Weldon survey

contains synopses of more than 50 studies, it can also be read as a study in research methods. Of the studies that are extensively quoted, two are field studies, and all others are laboratory experiments. The number of participants in the laboratory studies varies from 7 to 90, but it is not always specified. However, a small sample showed the number of participants was always listed in the original papers. The arrangement of the experiments was quite simple. Participants had to read a text, and correct errors, answer question on the text or execute some task described in the text. The duration of two experiments was mentioned explicitly. One experiment took two days, and the other two hours. From the descriptions of other experiments can be inferred that two hours is representative. In any case, no mention was made of experiments where longer term effects such as learning were studied. All studies focus on readability, which includes comprehension of text. More precisely, Dillon distinguishes reading speed, accuracy and comprehension as dependent variables. He also mentions several studies on fatigue. Presumably, the fact that the results of those studies were inconclusive was the reason why Mills and Weldon did not discuss them. The main independent variables studied were technical attributes of the text. As mentioned by Dillon, personal characteristics such as age and sex were not included as an independent variable in any of the studies. The exclusion of gender is, of course, surprising with regard to studies on colour. All experiments involved simple tasks and no experiment tried to simulate a real office environment with its customary noise and interruptions. According to Dillon, this resulted in a low ecological validity of the research. Another problem that was ignored was whether the subjects in the experiments were representative for the population of office workers who might be affected by the results of such studies, a subject that has attracted much interest in management game studies (Hughes and Gibson 1991). The typical research study involved the execution of two or more tasks by about 20 subjects under different conditions. The studies with the higher number of subjects split the population in groups and gave different tasks to the subjects in each group.

In the classification proposed by Dillon, most of the research covered by Mills and Weldon studies *basic ergonomic issues*. Newer research on these issues suggests that results in this area may be obsolete because of the continual improvement of visual displays. Otherwise, newer research concentrated on

manipulation facilities, which includes the uses of screens that have no direct equivalent on paper, such as text searching or hypertext. Dillon remarks that empirical data on advanced applications are not yet available.

This impression is not encouraging. If laboratory research would become important in a field like information systems development, one would at least hope for a better accordance between laboratory and field research, more imaginative experiments and consideration of wider range of independent variables.

3.3.3 Tables and graphs

The *tables and graphs* field can be subdivided into two fields, comparing tabular and graphical information and comparing different types of graphs, such as line graphs, bar graphs, and pie charts. As a subdiscipline of MIS, it started with the work of Scott Morton (1971), who studied the success of the introduction of a new management information system. That study can be classified as a field test, because it compared results of the new and the old system, but also as a case study, as it was limited to a single organization. It was also a success story that gave impetus to the DSS discipline and its descendants. Benbasat and Schroeder (1977) investigated the use of tables and graphs in the context of the Minnesota experiments (Dickson, Senn and Chervany 1977), a series of experiments where business students or middle managers played a specially designed simulation game. In the same vein, Lucas (1981) experimented with a logistic game. Research in this area obtained a quite high standing, culminating in the triple A score of Benbasat and Dexter with articles in MIS Quarterly, Management Science, and Communications of the ACM (Benbasat and Dexter 1985;1986; Benbasat, Dexter and Todd 1986a; 1986b) that investigated the relation between use of colour, graphics use, personality type and case type on the one hand and profits, time used and number of requests used in a simulation game on the other hand. Meanwhile, Ives (1982) noted the contrast between the rhapsodic claims of vendors and consultants and the meagre results of research, a theme that is echoed until this day. Both the Ives paper and the survey by DeSanctis (1984) called attention to the many articles on the subject from other disciplines, going back as far as 1926. Because these surveys showed no clear results, more

experiments were started, usually in limited laboratory settings, and with more independent variables. The newer research was surveyed by Jarvenpaa, Dickson and DeSanctis (1985), Montazemi and Wang (1988), Hwang and Wu (1990) and Vessey (1991) and after that, nothing exciting has happened in the field.

The general impression is that graphs are somewhat better than tables, but not much. The contrast with the increasing use of graphics can be explained in various ways. One view is that the increasing rigor of scientific methods applied in experiments guarantees the research findings are right, and that graphics users are merely the dupes of the salesmen of graphics packages. In this view, research should proceed in the current direction to establish some details that are not completely known. The opposite view is that the real advantages of graphics cannot be found by present research methods, which implies that the research strategy should be changed, for example by the use of field studies, or the research topic should be changed by an increased emphasis on the effect of presentation modes on persuading people (Garceau, Oral and Rahn 1988) or on getting attention from observers. A third view is that research studies that deliver generally applicable results are no longer appropriate, because with current PC software local field tests can be executed at low cost. In my view, business graphics serve real needs, and hence, if generally applicable research is done at all, it should be directed at new topics.

3.3.4 Effectiveness of decision support systems

Experimental research on the effectiveness of Decision Support Systems (DSS) started from the same roots as research on graphs and tables. The design of an experiment of this type seems simple. A number of representative subjects play a management game or simulation game. Half of the subjects are equipped with DSS type A and the other half with DSS type B. If profits of subjects with DSS type A are better than profits of subjects with DSS type B, type A is a better DSS. In experiments involving complicated games, subjects cannot be examined twice under different circumstances because the learning factor would presumably override all other independent variables. Because playing a realistic management game usually takes a whole day, research is costly. Moreover, both the gaming environment and

the DSS must be thoroughly tested before the experiment begins, because results before and after a change may be incomparable. Theoretically, it is highly probable that results will be inconclusive because the number of uncontrolled independent variables is so large. Moreover, because players must have freedom in decision making, it is not certain whether the differences in the DSS's, the program packages used by the players, are reflected in the dss's, the formal and informal information systems actually used by the players (Huber 1980).

The early research was adequately reviewed by Sharda, Barr and McDonnel (1988). Van Schaik (1988) demonstrated the pitfalls of this type of research. To increase external validity, he used managers rather than business students, but this limited his number of observations. On the other hand, his subjects unwittingly played against fixed scenarios instead of real opponents, which excluded noisy interactions, but diminished external validity. His results were inconclusive, which led trade papers to jokingly report that a DSS vendor had offered to buy all copies of his thesis because "he proved that DSS had no effect". The inconclusive results of DSS laboratory research may have led Guimaraes, Igbaria and Lu (1992) to execute a field study, but they did not extensively compare the two research strategies.

The decline of research on individual decision support systems coincided with the waning of interest in DSS proper. Newer experiments concentrate on Group Decision Support Systems, and focus on the way decisions are made rather than decision results (Benbasat and Nault 1990; Dennis, Nunamaker and Vogel 1990; Vogel and Nunamaker 1990). In this field, experimental research even seems to have preceded theory formation (Rao and Jarvenpaa 1991), which may be attributed to the need to justify expensive decision rooms.

My conclusion is that experimental DSS research suffers from uncertainty about the theoretical base of the field and a continuing shift in research interests. The theoretical base of this type of research was criticized by Vennix (1990) because of the complexity of the relation between the computer model supporting a decision and the effect of the decision itself. In his view, computer models influence *mental models*, mental models affect *policy decisions* and decisions induce *policy effects*, and those relations should be studied separately. He also showed the difficulty of assessing the relations between mental models and decisions and between decisions

and effects. The shift in research interest is demonstrated by the increasing attention for GDSS. Moreover, the DSS field is beset by a continual change in terminology. For example, much of the research that was done in the seventies under the name of DSS could now be repeated under the executive information systems (EIS) label. Finally, a problem in DSS research that has also been noted in the human-computer interaction field is that laboratory and field research seem to inhabit different worlds that are only united in surveys.

3.4 Conclusions

3.4.1 Games and Laboratory Research

Section 3.2 shows the rich research potential of games that can be used to simulate a wide variety of environments. Actually, this potential has not been fully tapped because some types of games, such as adventure games, have scarcely been used for education and research, and management games have been applied to a fairly restricted domain. Particularly, there are no reports of management games specially built for research and education in information systems development. Moreover, the broad range of topics addressed in simulation-gaming has not been reflected in the area of computer games discussed in section 3.2.

On the other hand, section 3.3 shows that the techniques actually used for laboratory research in information systems have been quite simple. One possible conclusion is that the scope for laboratory research in information systems is limited. This view would be supported by the decline of experimental DSS research with elaborate management games after the initial success of the Minnesota experiments. It would point to the abandonment of laboratory research in the field, probably in favour of field research. A discussion of this issue will be postponed to chapter 6.

A more balanced view is that, on the one hand, simple experiments suffice for research on human-computer interfaces, and that, on the other hand, laboratory research on decision support systems has failed to produce decisive results because the number of variables in the experiments has been too large. According to this view, laboratory research on information systems should be built on better

conceptual models, which supports our strategy to study the information systems development process in detail.

The development of laboratory research can also be seen from a historical perspective. In the seventies, graphics systems were expensive, and hence management had to be convinced of their benefits by experiments. In the nineties, spreadsheet graphics are available to any manager, so research into the advantages of graphics can be considered as useful as inquiring whether the average American or European needs a car or a telephone. This view is supported by the current popularity of experimental research into group decision support, which helps to evaluate the need for expensive decision rooms. In this view, the growing apprehension about software costs and quality would be a valid reason for enhanced interest in laboratory research on those topics.

According to the views expressed in the two preceding paragraphs, it is worthwhile to study the feasibility of a management game tailored to research and education in information systems. This is done in two steps. Chapter 4 demonstrates that such a game can be built by a description of Infogame, and chapter 5 shows that it can be used by a discussion of experiments with Infogame.

3.4.2 Verification and Validation

Because they pertain to both chapter 4 and chapter 5, the issues of verification and validation are discussed here. By *validation*, we determine that a conceptual model adequately represents the system under study, by *verification* we ascertain the equivalence of this model and the equivalent computer program (Kleijnen 1995). Formal verification requires that the conceptual model can be described in mathematical terms. Two approaches are *program verification* (Hoare 1969; Apt and Olderog 1994) and statistical comparison with analytic results (Kleijnen 1995). Both are most suitable for basic program modules. For example, Apt and Olderog prove the correctness of a procedure that sums the elements of an array, and Kleijnen discusses statistical techniques to determine the quality of a pseudorandom number generator. Because no mathematical model is available for Infogame as a whole, such verification methods can only be applied to modules of

Infogame. Examples are a mathematical proof that events will be retrieved from the event list in the correct order, and a statistical proof that results are not distorted by the assumption that machine defects occur only between jobs.

Validation of a game differs from validation of a simulation. A simulation program should produce the same results as the real-world system that is simulated, but the most important criterion for validity of a game is that players *consider* it realistic and act accordingly. For example, in MAGEUR, players follow real-world pricing policies by showing a preference for consumer prices ending with 95 or 99. The treatment of such prices by the market model is irrelevant, whereas it would be crucial for the validation of a simulation model pertaining to the same phenomenon.

A game can be validated as a test instrument, as a learning tool or as a research tool. A game is validated as a learning tool by comparing results from players and non-players in some independent test. For example, Vennix (1990) found no correlation between participation in a policy modelling game and mental model parameters. Woltjer (1995) found that players in an economics game performed better in a test than non-players. To validate a game as a test tool, results in the game are compared with results in an independent test. However, results in the game may also reflect learning effects because players who perform better in the game may also have learned more from it. Wolfe and Roberts (1986) found some support for the external validity of a management game as a learning and test tool by comparing performance in a management game to career success indices five years later.

Validation of a game as a research tool should be based on the comparison of game results and real-world results in similar circumstances. In our case, it would require a field study into the relation between quality metrics in successive phases of the information systems development process. This study has not been tackled because it would have been a research project of the same order of magnitude as the Infogame project itself. Accordingly, the validity of the results reported in chapter 5 rests on the assumption that Infogame players behave like real-world managers, which in turn is based on the impression that players went to great length to increase profits. For example, in 1994 teams applied industrial espionage, sabotage and cartels, and much indignation was shown at illegal actions of competitors.

4 Infogame

4.1 Introduction

Infogame[17] is a management game that has been specially developed for research and education in information systems development. In many respects, its design was based on MAGEUR (see 3.2.7). The design of Infogame was started as a Tilburg University research project in September 1985 under the direction of J.P.C. Kleijnen, and it incorporated ideas proposed in (Kleijnen 1981). A preliminary description was given in (Casimir 1986). The first version was completed in December 1988. At that point, the development of the research project was halted for two reasons. First, Infogame was considered unplayable. However, the fact that over 300 students have played it successfully, casts a slight doubt on the validity of this argument. Second, the Infogame model does not agree with the management science view that decisions on the strategic and operational levels are strictly separated, which implies that events on the operational level are not relevant to strategic decisions. A formal challenge to this view is beyond the scope of this thesis, but it is refuted by many instances of top management involvement with such trivia as poison traces in mineral water, alleged wear from washing powder and unauthorized derivatives speculation. Applications of Infogame to education started in September 1990. They are described in chapter 5.

Infogame has been described with three different purposes. First, instructions for game administrators and players are described in the current versions of the *Game administrator's manual* and the *Users manual*[18], which are published separately. A summary description of Infogame from the users point of view is given

---

[17] Originally, the name "Infolab" was used for this game, but this name was claimed for a typical *computer laboratory* (see 3.1).

[18] Apart from a number of student editions in Dutch, three versions of the Users manual have been published by Tilburg University: Rev 1.2, December 1988 (Research Memorandum FEW 363), Rev 1.3 June 1990 (Reeks ter discussie 89.07) and Rev 2.1 September 1994 (Research memorandum FEW 676).

in section 4.2.

Second, a detailed description of algorithms and data structures in Infogame is given in the *Technical description*. This document is needed both for maintenance and to prove the correct operation of the game. However, like the contents of a psychological test, the technical description of a game should not be made public. Otherwise, players may be tempted to give more attention to the detailed rules of the game than to the environment that is simulated in the game. However, the production, marketing and labour models, which are pivotal to the Infogame model, are described in detail in sections 4.4.7 and 4.4.8.

Third, the main part of this chapter discusses the decisions made during the design of Infogame to show the relation between features in Infogame and functions needed for its application in information systems development research and education. According to Greenblat (1987), such discussions are too often omitted in game descriptions. Both decisions to include features in Infogame and decisions to omit features from Infogame will be discussed. Of course, only a small part of the unlimited set of features that could have been included in Infogame will be examined. Some extensions that may be included in future versions are discussed in section 6.3. Most design decisions were made at the start of the project. They were based on *conceptual requirements* and *operational requirements* on the one hand, and on *modelling resources* and *technical resources* on the other hand. Later changes were mainly based on *experience* collected when Infogame was played by various groups. We distinguish three groups of design decisions. *Model design* concerns the way the model is built, for example the amount of detail. *Interface design* affects the communication between player and computer, and *physical design* pertains to choice of programming language, data structures, etc..

Game designs have also been published by Gremmen (1989) and Coppieters (1990). Gremmen's SIER game is essentially a four-country macro-economic model where players make decisions on such items as taxes and government expenditure and all computations pertain to fixed periods. Its design reflects the maturity of macro-economic modelling as a discipline.

Coppieters describes his design of IALTA, a large-scale game for the Royal Belgian Defense College, in terms of objects. A special feature of this game is that

the role of lower level officers can be played either by human actors or by *decision automata*. Hence, the design of decision objects is central to the game. In comparison to Infogame, IALTA shows a far lower level of abstraction, and consequently its size and the necessary design and programming effort are an order of magnitude larger than those of Infogame.

The remainder of this chapter is structured as follows. After the description of Infogame in section 4.2, section 4.3 discusses requirements and resources, and the relation between those and the design aspects. Model design, interface design, and physical design are discussed in detail in sections 4.4, 4.5 and 4.6 respectively. Sections 4.4 through 4.6 contain some details that may not be of interest to the general reader. However, these are included to give a clear impression of the difficulties encountered in designing and implementing Infogame. A discussion of the value of Infogame as an experimental tool is postponed to chapter 5.

4.2 A brief description of Infogame

Players or player teams in Infogame manage a firm that produces and markets one or more products. Up to 12 firms compete in the market by product quality, price, and advertising budget. Players make production and marketing decisions for a game period that represents a quarter in the simulated world. When all players have entered their decisions, the game administrator computes the results of the round and transfers them to the players. Subsequently, players examine their results and enter their decisions for the next round. Time between rounds is determined by the game administrator and may vary from half an hour to a week. On a 1995 PC (90MHZ pentium), results are computed in a few minutes.

From the point of view of the player, two features distinguish Infogame from the conventional management game described in section 3.2.7. First, players are not provided with standard reports such as financial accounts and production and sales statistics, but with detail data that cannot be interpreted without further processing. Accordingly, players really need an information system for decision making. Second, some player instructions are executed in an indirect fashion. For example, by defining a *reorder level* and *order quantity* for a product, the player

actually states that an amount equal to the order quantity should be produced whenever the stock is below the reorder level.

A product is marketed in one of four industries. Competition is limited to products in the same industry. Each product is produced with a technology that determines the type of machine used, the production capacity, the number of employees needed to attain full capacity, the materials needed, and the product quality. Sales are made to simulated consumers who choose a product on price, quality, advertising budget, and expected delivery time. Whereas in other management games sales for a single product are aggregated, several hundreds of individual sales transactions per quarter are simulated in Infogame.

Three types of industry are implemented:

a) Industries with *production to order only*, where each consumer order is translated into a production order, that may be annulled when late.

b) Industries with *production to stock only*, where a sales transaction can only be made if the product is in stock. Production is controlled by a player defined *reorder level* and *order quantity*. Whenever the stock is below the reorder level, a production order for the amount specified by the order quantity is executed.

c) Industries with *production to stock with back-orders*, where production is controlled by reorder level and order quantity, but existing back-orders are filled first after completion of a production order. Consumer orders can be annulled if they are not met in time.

Purchase of materials is also determined by reorder level and order quantity. Average and standard deviation of delivery time can vary between suppliers, and production may be stalled because of lack of materials. Infogame can also contain an industry that produces semi-finished products for use in the end products of the same firm. Because a semi-finished product can also be used as a material for another semi-finished product, the flow of materials can become very

complex. A simplified picture of the flow of materials, without semi-finished products, is given in Fig. 4.1.

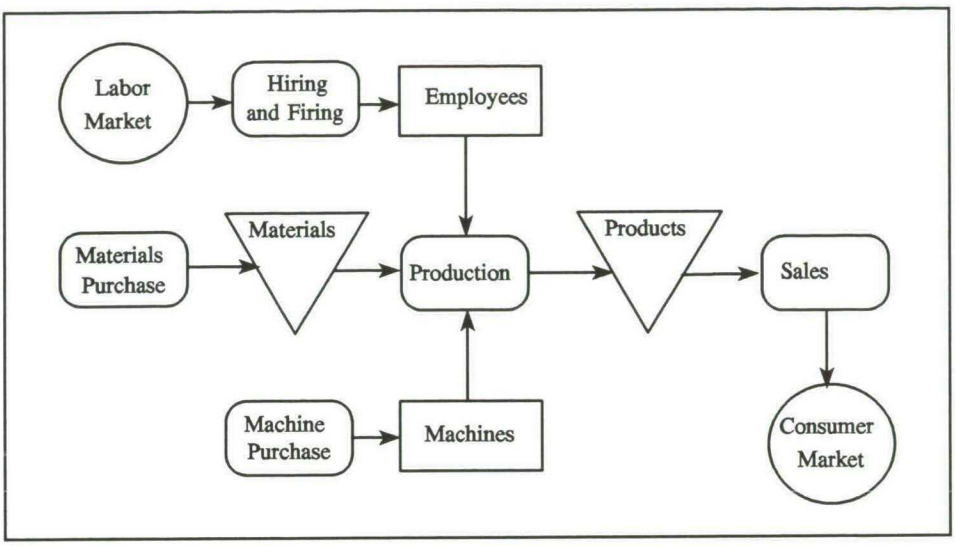


Fig 4.1: Flow of materials

Employers compete on the labour market for employees, who are simulated individually. Players must determine a minimum and a maximum number of employees. If current employment is below the minimum, employees are hired and if it is above the maximum, employees are fired. In contrast to production and purchases, the number of employees cannot be adapted to variations in demand during a quarter.

Player decisions that also occur in conventional management games are investment in machines, demand for loans, credit terms for customers, and acceptance of credit from suppliers. A graphical representation is given in Fig. 4.2, and a full list in Table 4.1. In the interface program, decisions can be grouped in various ways. For example, suppliers can be chosen in the same instruction as materials or in a separate selection instruction.

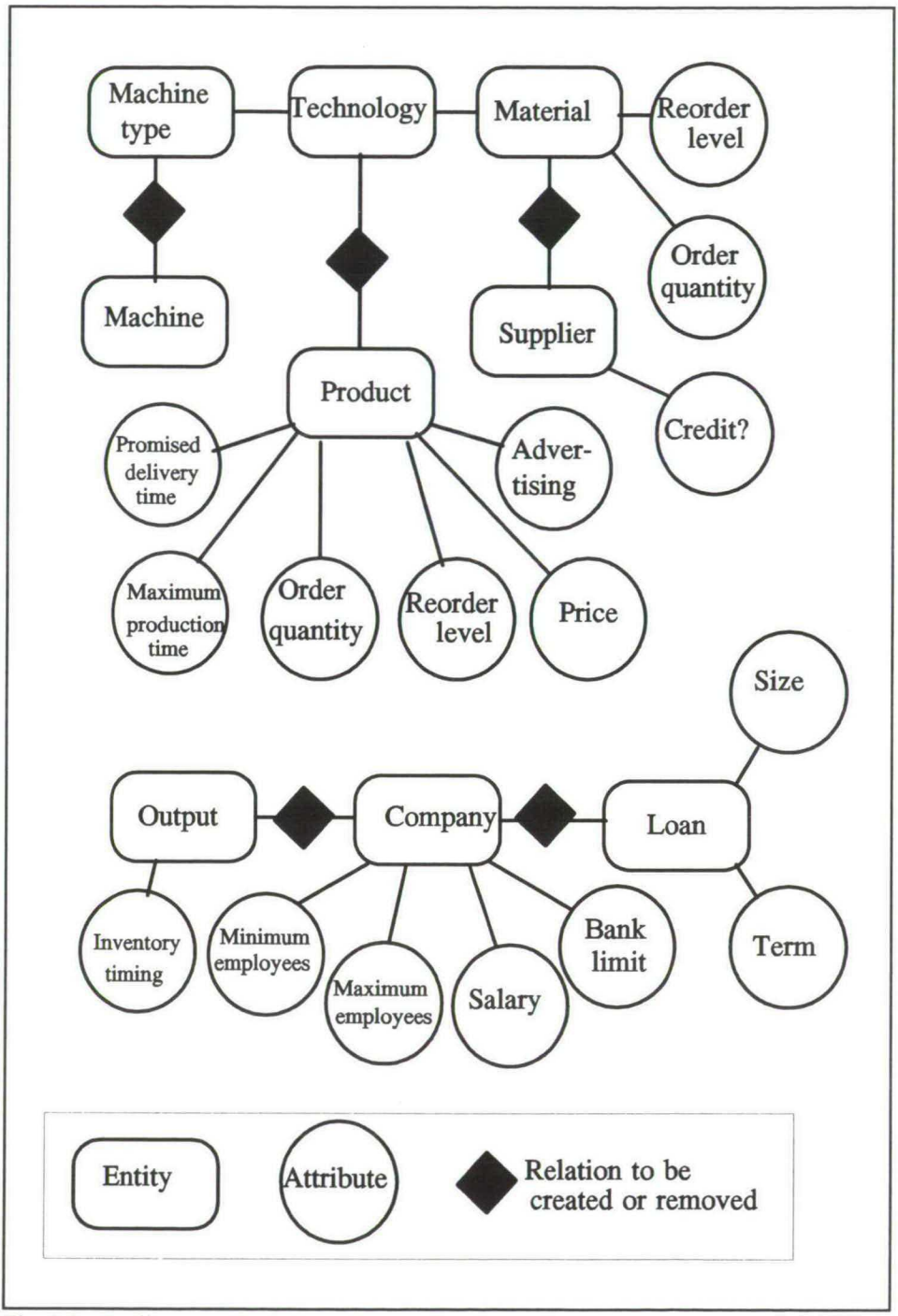


Fig 4.2: Attributes and relations that can be changed

Table 4.1: Instructions

Entity	Attributes
Output	Report used or not used, timing of inventorying
Machines	Invest or scrap
Product	Product: add or remove, technology: add or remove, price, reorder level, order quantity, advertising budget, promised delivery time, maximum production time
Materials	Reorder level, order quantity, suppliers, supplier credit
Employees	Salary, minimum and maximum number of employees
Bank	Ask loan or credit limit, repay loan
Industry	Days of consumer credit

A distinctive feature of Infogame is that the player decides what types of events will be reported. For example, a player who opts for cash sales has no need to separately record sales and sale payments. If a type of event is reported, every single event of that type is recorded in the report file. Apart from collecting event data, a player can collect inventory data at specified intervals. This allows the development of simple information systems that only monitor stocks and utilization rates. Because in reality, data cannot be collected free, the game administrator may attach a cost to the collection of each data item.

All aggregated data, such as quarterly sales, production or profit, must be computed by a separate information system, that may be supplied by the game administrator or developed by the player. Apart from this information system, Infogame contains an operational information system that computes the cash level to determine interest charges, and stock levels of materials and finished products to determine whether sales are allowed, production can be started or purchase orders must be placed. Variables in this information system, such as the number of unfilled sales orders, are not directly accessible to the player.

The game administrator can tune the game by determining the number, size and characteristics of markets, employees, materials, supplier, machines and production technologies. Usually, this also entails a change in the players' manual.

4.3 Requirements and resources

4.3.1 Introduction

For a successful project, requirements and resources must be balanced. Definition of requirements without consideration of resources leads to grandiose schemes that will never be completed. Concentration on resources produces technically perfect systems that never will be used. Flexibility is needed to adapt to changes in resources and requirements. For example, restrictions due to the small memory size or low speed of PC's should be lifted as soon as more powerful computers become available.

Requirements for Infogame are defined at two levels. The conceptual requirements define its purpose. Hence, as will be shown in section 6.3, they are influenced by new educational and research priorities. The operational requirements determine how it will be used. They will be altered to adapt to new player groups and changes in game organization. Accordingly, there is a fairly direct link between operational requirements and technical resources. For example, the progress of networking may engender a demand for network gaming.

Resources are divided into modelling resources, which embody the modelling knowledge of the designer, and technical resources, which concern the available hardware and software and programming capacity. In large-scale projects, technical resources are limited by financial resources, because hardware, software, and programmers time can be bought, but the development of Infogame was a small-scale project without a formal budget.

4.3.2 Conceptual Requirements

From the start, Infogame was intended as a *management game* that should be played with a *specially developed information system*. Accordingly, it should contain the features of a traditional management game, such as the existence of competition between firms on one or more markets, and the need for coordination

between production and marketing. The choice for a management game, rather than a simulation game without competition, was based on the notion that information systems can be used for competitive advantage. Moreover, a competitive game was thought to provide a more stimulating environment, which would induce more natural behaviour from players.

To increase the challenge of information systems development, it was decided that Infogame should be *complex*. To ease transfer of information systems development skills between Infogame and the real world it was also decided that it should be *realistic*. Moreover, the rules of a realistic game can be explained succinctly to players by comparisons with the real world. The case for realism in gaming is, of course, based on the assumption of a common background, which includes an appreciation for Western notions such as the benefit of competition and the importance of company profit.

To stress the importance of the specially developed information system, it was decided that Infogame would not provide any output that could be immediately used for decision making, such as financial statements. The output file would only contain detail data that needed further processing for use in the game. If an uniform information system would be needed, for example to compare player results, it could be separately developed. In contrast, related studies (Dickson, Senn and Chervany 1977; Courtney et al. 1978; Courtney, DeSanctis and Kasper 1983; Van Schaik 1988; Yeo and Nah 1992) employ decision support systems built on top of the accounting and reporting systems of a conventional management game.

4.3.3 Operational Requirements

The first operational requirement was that, once the requisite information system was built, it should be possible to play a game round in the time normally allotted to a management game round, say half an hour to one hour. Consequently, Infogame players should have no more decisions to make than players in a conventional management game. This requirement also limits the time to compute the results of a round to about five minutes. Initially, this influenced physical design to some extent, but with the speed of present-day PC's, it is no longer significant.

A second requirement was that players should be allowed to input their decisions in a simple way. This suggested that it should be possible to adapt the input program to specific user needs. Moreover, all decisions would have to be entered at the start of a game round. This required a method to specify decisions to be made during a round.

4.3.4 Modelling Resources

Both the background in simulation of the original sponsor and my experience with model building in conventional management games led to the choice of a simulation model to create a gaming environment. The alternative of using more abstract mathematical models, such as systems of simultaneous equations or linear programming, was not studied in depth.

On a lower level, we used scheduling models, known both from operating systems (Brinch Hansen 1973) and manufacturing planning and control (Vollmann, Berry and Whybark 1992), and the IIasa model (Anthonisse, Van Hee and Lenstra 1987). Because we intended models to be adequate, rather than optimal, no extensive study of management science models was made.

A more detailed study of planning models at the start of the design would have had some advantages. First, it could have resulted in the incorporation of alternatives to the independent demand function, such as MRP (Vollmann, Berry and Whybark 1992). Second, it could have resulted in a better balance between time-driven activities, such as weekly planning schedules, and event-driven shop-floor activities.

4.3.5 Technical Resources

Technical resources were considered within the strict limitations of the project. Standard PC's (originally IBM-XT's) and a Novell network had to be used, and all programming would have to be done by myself in a programming language that was available within Tilburg University. Though such tight restrictions might seem a severe handicap, they also precluded experimentation with unknown

hardware and software and prevented an unwarranted emphasis on technical accomplishments. Moreover, flexibility was enhanced by the fact that design and programming were in the hands of one person. Surprisingly, apart from the increase in memory capacity and speed of the entry-level PC by one or two orders of magnitude, no major technical upheavals have occurred in the PC-world in the 1985-1995 period. Although it is often criticized, MSDOS is still widely used as an operating system, Novell Netware has increased its hold of the LAN software market, and no programming language has emerged that is both generally accepted and definitely superior to Pascal. Increased capabilities of the standard PC allowed the discontinuation of the Infogame version for a PC with diskettes only, and decreased the original preoccupation with speed.

4.3.6 Experience

There were three reasons for changes in Infogame as a result of experience. First, data deemed necessary by players, such as data on prices and sales of competitors, were absent from the output files as originally defined. Second, some model features had unlikely results. For example, it might be impossible to secure an adequate labour supply. Third, some complicated rules where difficult to understand for players, especially because they did not agree with real-world practice. For example, machine scheduling was difficult because a production technology might use more than one type of machine. Finally, when Infogame was actually played by a large number of students, protection against sabotage, espionage, and incorrect inputs appeared to be necessary. This led to a change in the transfer of input and output between players and game administrator. The six types of changes resulting from experience discussed here give a fair impression of the pitfalls of game design.

Introduction of additional reports

In the first versions, players could not compare prices or compute market shares because the report to a company contained only data on its own transactions. In later versions, market report files with data on prices, advertising budgets and

sales of all products in each industry, and a labour file with data on the number of employees and salaries of all firms are supplied to players. This is in line with the emphasis on external information in executive information systems (Turban 1993).

In the original version, the game administrator could only consult the files produced for players. Consequently, no standard reports, such as financial accounts, could be provided because the necessary data would not always be available. To avoid this problem, in most experiments all data were provided free of charge to all players. Because it is important that players decide which data to select, a separate file with all data is now produced for the game administrator. The reason for this correction was that in the original design, the separate information requirements of the game administrator had been neglected.

Generalization of the sequence number concept

In the first version, an order number was included in some, but not all production records to trace the execution of production orders. However, when some information systems were being developed, it was found that sequence numbers could be useful in other records too. For example, sequence numbers for sales and purchase orders can be used to compute delivery times. Accordingly, in later versions a separate field for sequence numbers was added. This type of change cannot be prevented because the full use of the game cannot be foreseen. It can be compared to the addition of new features in programming languages.

Change of labour market

In the first version, players defined the number of employees to be hired or fired. Companies that neither hired nor fired employees could lose too many employees because leaving employees were not replaced unless employees were hired. In the present version, players determine the minimum and maximum number of employees. The original labour model was also needlessly complicated because employees entered or left a firm only one or more months after being hired or fired. In the new version, firing is discouraged by a redundancy payment. In this case, the

challenge of implementing a complicated model led to a neglect of the requirement that the model should be understood by players and provide predictable results.

Simplification of technology definition

In the first version, a technology could employ more than one type of machine, and more than one unit of each type. This notion, which was derived from operating systems theory (Brinch Hansen 1973) did not conform to industrial practice and it caused intricate planning problems. In the new version, each technology uses a single machine. This change was simple, but some work would have been saved if the new specification had been implemented in the original version. This stresses the importance of a thorough study of real-world items simulated in a game.

Annulment of orders

In the original version, late back-order deliveries were refused at the moment when production could be started. As a result, the market could be upset by waves of nondeliveries, and back-orders could remain in the system infinitely. In later versions, orders are annulled at the moment they are considered late. In this case, the original model was based on the incorrect assumption that it would be technically infeasible to cancel planned events. Actually, model requirements should have dominated technical restrictions from the start.

Change of data transfer

Originally, the main program and the interface program shared data structures, and main program files were accessible to the interface program. Input data from the interface program were written to an ASCII network file. To create a better fence between main program and interface program, output data from the main program to the interface program are now also written to an ASCII file. This change was due to insufficient attention for operational aspects in the design phase, and to unpredictable changes in the computer environment.

4.3.7 Requirements, resources and design decisions

The relation between requirements, resources and design decisions is illustrated in Fig. 4.2. It should be stressed that, though technical resources limited the extent of the model, physical design did not influence model or interface design. Although this conforms to theory, in practice models are often changed because computational requirements of the original model cannot be met with the chosen combination of hardware and software, for example when fourth generation tools are used with standard PC's.

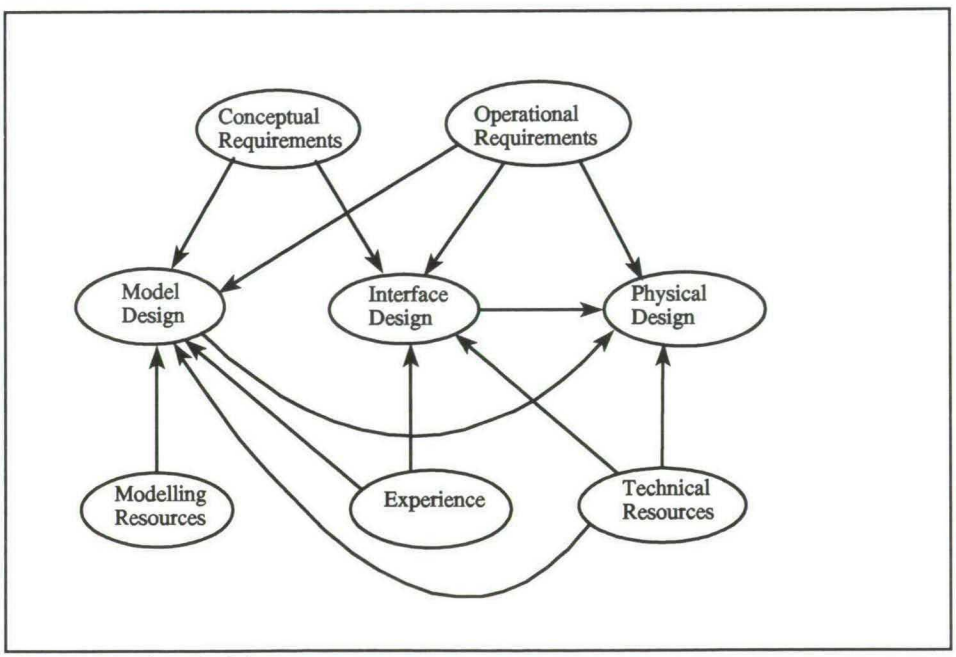


Fig. 4.2 Design decisions

4.4 Model Design

4.4.1 Introduction

The choices made during the design of the Infogame model can be approached from two directions. First, design variables that are common to all games, such as *timing*, *randomness, information*, and *abstraction level*, can be studied. Second, three types of design decisions can be distinguished:

a) What *topics* should be included in the model
b) What *details* should be included in the model
c) How should specific *aspects* and *parts* of the model be built

The first decision that had to be made was whether Infogame should be played as a real-time game. Technical arguments against the design of a real-time game were the large effort needed for implementation and the instable network environment. An educational argument was that, in our opinion, Infogame should emphasize the design and use of strategic, rather than operational information systems, whereas in a real-time game operational decision making would predominate. Accordingly, it was decided that Infogame should be played in rounds. Other timing decisions are examined with the level of detail in section 4.4.3.

Design decisions of different types are interrelated. For example, the decision to incorporate labour relations in the model is connected to the decision to use a detailed labour model. There is no difference between the choice of topics in Infogame and the choice of topics in a conventional management game, but the level of detail in Infogame is defined differently. In a conventional management game, players must directly deal with details, but in Infogame, detail actions are handled by programmed rules, and detail reports are processed by the information system.

The remainder of this section is structured according to the second option. The choice of topics, which is related to the level of abstraction, is discussed in section 4.4.2. The level of detail and the degree of randomness are examined in section 4.4.3. Section 4.4.4 addresses the *entering of decisions*, and section 4.4.5

deals with *information available to the player*. As a specific aspect, *Start-up and end-of-game effects* are addressed in section 4.4.6. Two specific parts of the model are considered in depth: the *production/marketing model* in section 4.4.7 and the *labour model* in section 4.4.8.

4.4.2 Topics

4.4.2.1 Topic selection

Three approaches can be used to determine the topics that should be incorporated in a management game, the *empirical* approach, the *educational* approach and the *modelling* approach.

The empirical approach

According to the empirical approach, players in management games should take the decisions that are usually taken by top managers. One objection to this approach is the difficulty of obtaining reliable data on actual top management concerns. For example, empirical research has not definitively dispelled Parkinson's (1958) "law of triviality", that states that the time spent on any item on the agenda is inversely related to its importance. This example also shows another objection to this approach. Game-playing is often used to improve management practice, and not to foster prevailing habits. For example, a game focusing on information systems may be used to increase top management attention for this subject, precisely because surveys have shown little management interest in information systems.

The educational approach

If the educational approach is followed, the game designer determines what topics he wants to teach and incorporates those topics in a management game. However, the fact that knowledge of a functional area is important does not imply it should be incorporated in a general management game. Such an area might better be

taught by functional games, case studies, or other means of instruction.

The modelling approach

We prefer a third approach, which we call the modelling approach. We start from the *core area*, one or more functional areas that are selected by the educational or empirical approach. A functional area not in the core area is incorporated in the game if and only if decisions in this area are interrelated with decisions in the core area. A functional area is not incorporated if optimal decisions in that area can be made without regard for decisions in other areas. This entails that all processes that can be effectively handled by decision automata in the terminology of Coppieters (1990), are considered as black boxes in the game program. For example, if the determination of the advertising budget is in the core area, allocation of the budget to different media is incorporated in the game if it affects other marketing decisions, such as distribution, and it is omitted if it is assumed that an optimal allocation to media can be made without regard for decisions outside the realm of advertising.

4.4.2.2 Abstraction level

The abstraction level of Infogame is set fairly high. To better appeal to students' intuition, Infogame firms are described as manufacturing companies in an unspecified industry. However, the model can also be used for service companies because a *service* can be defined as a product that cannot be held in stock. The game administrator can opt for a lower abstraction level by an appropriate change of description and parameters.

4.4.2.3 Topics in Infogame

The core area of Infogame consists of marketing and production. We think that the dominant strategic business issues are the choice of market and technology, and that market and production information is needed to support decisions on those issues. Moreover, information systems are especially useful for the coordination of

production and marketing, a subject that will be discussed in detail in section 4.4.7.

On the production side, investment in machines, purchasing, and labour relations were introduced. Investment in machines is a regular topic in conventional management games. Both purchasing and labour relations were introduced to provide a realistic model of the production process. Machine breakdowns were incorporated as an example of an operational detail that may have results on a higher level, for example by increasing delivery times, and is dependent on strategic decisions, because the choice of the machine determines the probability of a breakdown. However, the decision to introduce machine breakdowns without providing the means to influence their occurrence by maintenance policy may be questioned.

Consequently, from the main problem areas in production management, explicitly listed by Zimmermann and Sovereign (1974), but also found in similar texts (Buffa 1983; Schroeder 1981): *capacity, plant location, plant layout, capital budgeting, aggregate planning, inventory, scheduling,* and *maintenance*, only plant location is definitely absent from Infogame. The primary reason for this decision was that our resources did not allow the programming effort needed to implement a real or fictitious geography. For example, the graphical interface of IALTA (Coppieters 1990) consists of 35000 lines of Turbo-Pascal code and its development took 60 man-month. Moreover, we do not think that geography is as important in management games as in military games. There is even a danger that it could distract users, and it would generate a demand for specialized geographic information systems that cannot be built by the average information systems designer.

From the famous four P's of marketing (Kotler 1986), we included *product quality*, *price* and *promotion*. Promotion was limited to setting the overall advertising budget, because media choice was not considered a top management responsibility. Customer credit was added because it is a typical subject for clerical data processing systems. Distribution was omitted because its inclusion would have entailed simulation of retail stores with independent information processing systems. This would have taken too much programming effort and it would have made the game too large for the hardware used when Infogame was designed. It was also thought that this would have made the game too complex for players. Physical distribution was omitted for the same reason as plant location. Moreover, though the importance

of logistics has been recently emphasized, it is often entrusted to specialized firms.

The financial model in Infogame is simple, because we did not see a direct link between the core area and activities such as share transactions, intercompany loans, joint ventures, mergers, sales of assets, and other deals that dominate the financial high-tech world. Accordingly, we think those issues are best addressed in a conventional management game. However, players with experience in other games often asked whether Infogame offered opportunities for mergers and stock issues, and adapting an information system after a merger certainly is an interesting subject for an experimental study. Accordingly, mergers may be included in a future version of Infogame.

Apart from competition in the labour and consumer markets, interaction between players was not included in Infogame. For example, a company that produces materials cannot sell those to other companies. The main reason for this restriction was the notion that players should not invest too much time in contacts with other players. However, because of the increasing interest in interorganizational information systems, trading among player managed firms as well as sales to simulated firms might be implemented in future versions.

According to the original specifications, innovation by introduction of new technologies was to be part of Infogame. However, later it was decided to use a fixed set of machines and technologies. A reason for this change was that the game administrator should be able to adapt this set, and that this task might become too difficult if it also contained a dynamic element. Moreover, it was not obvious how the additional information on the environment should be conveyed to players after each round. Because the Infogame environment has now been stabilized, those objections are no longer valid, but the implications of the introduction of innovations must be thoroughly studied before implementation.

4.4.3 The level of detail

In the real world, most companies sell a large number of products, and each product is produced and sold many times. In a conventional management game, a company produces and sells a few products, and for each product, aggregate sales

and production for a year or a quarter are determined. Infogame models a company in a more detailed fashion than a conventional management game, but it still represents a simplified model of the real world. Because we decided to use a simulation model, we could choose between two models to simulate the events occurring in a single period:

a) An *event-driven simulation* model that sequentially simulates a large number of individual sales and production transactions for a few products.

b) A *time-driven simulation* model that simulates aggregate production and sales for a large number of products in one or more distinct time-buckets.

In event-driven simulation, the interaction between transactions is straightforward. Each event is influenced by prior events. For example, a product cannot be delivered if a previous delivery has exhausted the stock. Any conflict between events that occur at the same moment is resolved by the event scheduler. In a time-driven simulation, there is no clear causal chain. For example, if total production capacity is 100,000 units and production plans for 1000 different products add up to 120,000 units, it is not clear what plans should be reduced.

Event-driven simulation of sales and production was chosen for three reasons. First, the detailed reports supplied to players are interpreted more naturally when events can be timed like real world events. This is important, because instruction manuals should describe only in which respects the simulated world differs from the real world. Second, it is difficult or impossible to analytically compute the results of interactions among different types of events. This is shown by the limited results of analytic queuing theory, which is applicable only to a few simple models. Moreover, mathematical models tend to study equilibrium states, whereas we need data on states after a specified number of events (Gross and Harris 1985). In general, analytical methods such as queuing theory are used as an alternative to simulation, rather than a technique within simulation modelling. Third, we think this model poses more challenges for information systems developers. For example, it is no trivial task to build a time-driven view to the event-driven model.

The model of the labour market is time-driven. Each Infogame firm employs a large number of identical workers, who, according to European custom, are hired for a quarter or longer. It was decided to deal with each worker individually, especially with regard to hiring and firing, but no individual differences between workers were implemented. This is one topic where physical restrictions have somewhat influenced model design. It was assumed that the total number of workers in the game should be allowed to grow to several thousands, and consequently the infamous 640 K barrier of MSDOS, which only recently has been lifted by Borland Pascal 7.0, limited the number of worker attributes.

Random numbers are used in the infogame model to determine the behaviour of individual customers and workers. For example, the time between two purchases of a customer, the amount he will buy, the probability that he knows a product and the product that is ultimately chosen are all drawn from a random distribution. However, randomness is not the only factor that creates uncertainty about the market, because prices and advertising budgets of competitors are not known either. In the production model, the only random factors are the chance that a machine fails and the repair time. A production order is completed at the planned moment, the amount produced is exactly equal to the amount ordered, and actual resource use is equal to planned resource use. Accordingly, there is no need for a management control system that determines whether plans have been executed properly. The design of an Infogame version where management control systems can be applied is described in section 6.3.4.

4.4.4 Player decisions

In the real world, decisions are either directly executed, for example by signing a contract for the delivery of a machine, or they are transmitted to operators or lower-level managers, for example by stating the amount that must be produced. In conventional management games, all decisions are directly executed insofar as allowed by the prevailing restrictions such as production capacity. In Infogame, decisions on price, advertising budget and investment are executed directly, but this is not possible for purchasing and scheduling decisions. In the real world, those are

made daily by operators or low-level managers using current data on such items as stock levels and open orders. Because the necessary data are not available, those decisions cannot be made by Infogame players at the start of a quarter. On the other hand, because Infogame is not a real-time game, they cannot be entered at a later moment either. Accordingly, players must define rules that will be observed during the simulation of the game period. Because the circumstances that will prevail when a rule will be used are not known when it is defined, rule definition is difficult, as is well known to rulemakers such as legislators, programmers, composers, and authors of manuals. Rules are, however, indispensable if decision makers want to control events when they are not on the scene. Accordingly, rules are pivotal to the design of Infogame. We distinguish three types of rules. In the present version, only the first two types of rules are implemented. The third type is discussed in section 6.3.3.

a) Fixed rules, for example: *Production cannot start unless sufficient materials are available.* The player must know such rules because they influence the results of his decisions, but he cannot change them.

b) Rules with player defined parameters, for example: *An order for a player defined quantity of a material is placed when the stock falls below the player defined reorder level.* Production, material buying and reporting are controlled by such rules. Accordingly, many player decisions in Infogame define rule parameters.

c) Rules defined by the player, that are formulated in some programming language, for example an expert system shell, such as: *IF production < 0.8 * capacity AND stock > 0.5 * sales_last_quarter AND price > 1.4 * cost_price AND price > average_price THEN SET price TO 0.9 * price*

4.4.5 Information for players

Infogame provides no aggregate reports, but only raw data on events. For decision making, the player has to make or buy an information system, because the uncondensed report is unsuitable for that purpose. Infogame produces ASCII output files with fixed format records. The interpretation of a field depends on the type of record. For example, the same field indicates the number of products sold in a sales record and the amount paid in a customer payment record. A record either reports an event, for example a sale, or a state, for example the size of the stock of a product. The player determines what types of events and states are reported, and at what intervals states are reported. In the present version, where all plans are executed (see 4.4.3), redundant information is given if all reports are requested. For example, the size of stocks can always be computed from production and sales data. Only states that could be directly observed in the real word may be reported in Infogame. For example, the number of employees working on a specific order may be reported, because it can be counted in the real world, but the number of unfilled orders may not be reported, because, in the real world, it can only be determined by the use of an information system, and the design of information systems is the task of the player.

The ASCII format was chosen to postpone the choice for a programming language or application package. In our experiments, Infogame output files have been read by Pascal programs and imported into databases and spreadsheets.

4.4.6 Start-up and end-of-game effects

In the majority of management games, players start with going concerns, but in Infogame, all companies start from scratch. This forces players to take decisions such as choice of market and technology, which are intimately connected with the design of the information system. For example, a player who chooses to operate in a market with production to order must control delivery times, whereas a player who produces for stock must monitor his stocks. In a game that starts with going concerns, all players must initially design information systems that are suited

to the starting conditions. Subsequently, any major change in policy requires a change in the information system, whereas players who refrain from radical changes may attain satisfactory results with far less effort. Because many participants in experiments may prefer the latter course, much of the diversity of the Infogame environment would be lost. However, a game administrator who prefers a start with going concerns can execute one or more rounds with a fixed scenario. In some of our experiments, we have executed four rounds (one year) according to a fixed scenario to provide players with historical data. In those experiments the options for subsequent quarters have also been restricted to suppress the need for a radical redesign of the information system.

Like other management games, Infogame may suffer from the end-of-game effect that occurs if in the last round players take decisions that may improve their score in the game, though they are not in the long-term interest of the firm. For example, research and development are decreased, market research is eliminated, and stocks are sold off. In Infogame, such actions are countered by a scoring method based on the going-concern value of firms.

4.4.7 The production and marketing models

Infogame uses two production models. In industries with production for stock, goods are produced in fixed-size batches. A production order is given whenever the stock is below the reorder level. Apart from the size of the batch, the player can also specify the time allowed for production of a batch, but it is not possible to specify that orders should be given at specific moments, such as the start of the week. In industries with production to order only, a production order of the same size is automatically given for each consumer order.

In both models, all production orders are entered into an order queue, and they are executed according to a First-Come-First-Served algorithm when all necessary production factors, i.e. machines, labour and materials, are available. On the lowest level, some simplifications have been introduced. For example, during executing of a batch, machines do not fail and the number of operators remains constant, even if the number of employees changes during execution of a batch. The

motivation for these simplifications is discussed in section 4.4.8.4. The main defect of this model, that simulates shop-floor procedures that are current in many industries, is that it offers no scope for production planning in time buckets instead of lots, as is customary in MRP planning (Vollmann, Berry and Whybark 1992).

In the marketing model, individual consumers are simulated. Each customer is characterized by an index number, which determines both the maximum price he can pay and the maximum quality he takes into account. Each time a consumer comes to the market, he selects one of the products that are known to him, either because of advertising, or because he or a consumer with a similar index has bought it recently. If production is for stock only, a product is only considered if it is in stock, otherwise, the delivery time should be acceptable. If delivery of an order takes too long, the order is annulled, and the consumer selects another product.

There were several reasons to select this detailed model of the buying process instead of a global model based on marketing theory. First, processing of detail data offers more scope for creativity in information systems development than handling global data. Second, the model can be understood and even built without marketing knowledge, which makes it suitable for a wider audience. Third, the model is flexible because it can be extended without essential changes in the remainder of the model, for example by introducing more than one quality index. Finally, some marketing features, such as market niches, can only be modelled by segmenting the market.

Though apparently a market model that simulates individual consumers requires the processing power of modern computers, the simplicity of the model is emphasized by the fact that Andlinger (1958) introduced individual consumers in the McKinsey Business Game to permit manual computation of market shares.

The interaction of production and sales is shown in the following example. A trader sells a single nonperishable product. Customer demand is a Poisson process. Every night the stock is replenished to the reorder level. Total sales and average stock during a year may be computed for various reorder levels by simulation. From the simulation literature (Kleijnen and Van Groenendaal 1992), it is well known that this problem can be solved either by repeatedly drawing the time of the next sale with the negative exponential distribution (see Fig 4.3), or by

drawing the total number of sales from the corresponding Poisson distribution (see Fig 4.4).

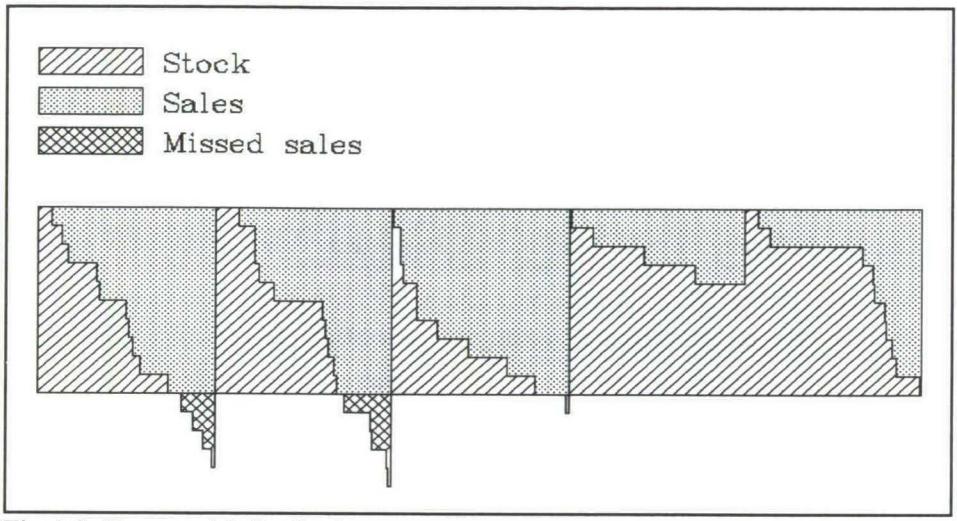


Fig 4.3: Exponential distribution

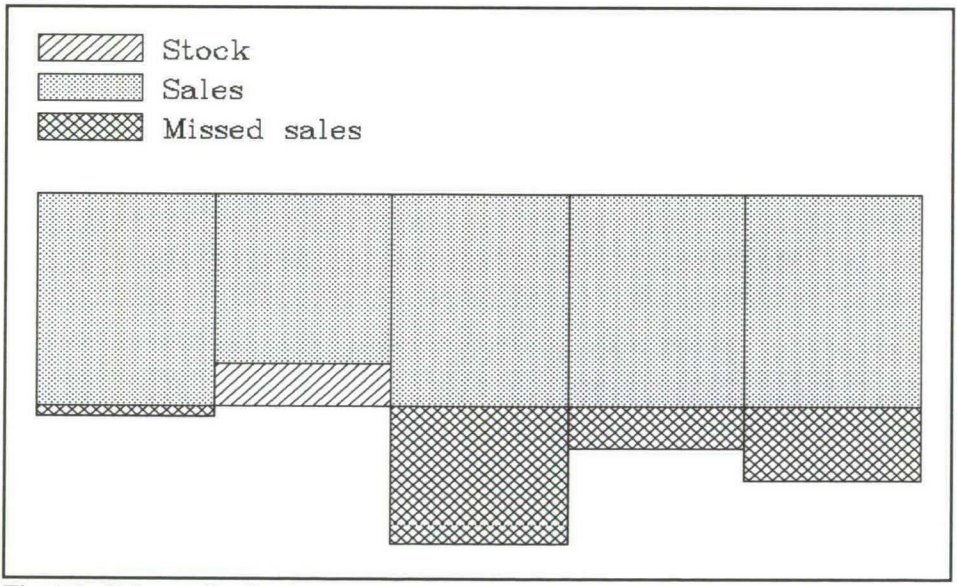


Fig 4.4: Poisson distribution

If, however, we assume that supply is also a Poisson process, the first method can be adapted to the new specification without any difficulty, as shown in Fig 4.5, but the second method is not applicable without considerable effort, because an analytical distribution for the cumulative results is not known to exist (Winters and Hendrix 1986).

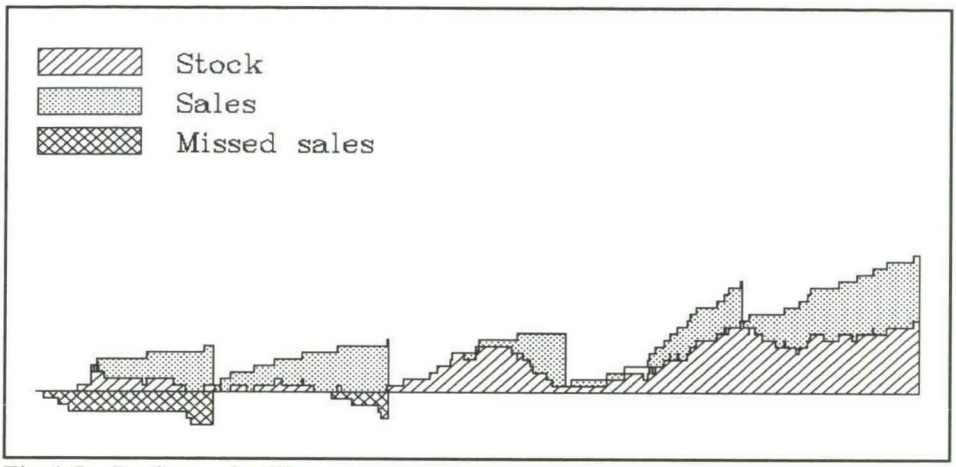


Fig 4.5 : Buying and selling

This example shows that analytical distributions for all interactions in Infogame, which usually involve three or more independent processes, can only be found by a major effort in operations research, which is not our main objective. Moreover, from a modelling and programming point of view, event-driven simulation is an established technique (see section 4.6.4).

4.4.8 The labour model

4.4.8.1 Introduction

Labour relations are explicitly implemented in Infogame because in most industries, labour is the most important production factor. When the decision to implement a labour model was made, it was also decided that employers should compete on the labour market. Whereas in the consumer market model a seller is

chosen for each sales transaction, employees are assumed to remain with their present employers unless they explicitly apply for a job with another employer. The number of persons in the labour market decreases when unemployment is high, and it increases when there is a high demand from prospective employers. No distinction was made between different classes of employees, because this would make the model too complex.

In contrast to other submodels of Infogame, the labour model is time-driven rather than event-driven. Workers start, change, and terminate jobs only at the start of a month. This is done because of the computational complexity of the labour market model, where a large part of all workers consider jobs on offer. It is also in accord with Dutch practice, were workers are hired by the month.

According to European custom, hiring and firing employees is not easy. In the first version of the model, employees could only be fired after a term determined by the player and they only could switch to another employer after the same term. However, this rule was difficult to explain to players and its results were unpredictable. For instance, an employer who did not hire any personnel could lose too many employees. Moreover, the rule did not acknowledge the fact that, in Infogame, decisions to hire or fire employees are always given in the first month of a quarter. Therefore, in the present version of the model, players must specify the minimum and maximum number of employees instead of the number to be hired or fired. As long as the current number of employees is lower than the minimum specified, employees are hired. If the current number is higher than the maximum, the superfluous employees are made redundant. However, they are only discharged at the start of the third month of the quarter. Because redundant employees have a higher propensity to apply for other jobs, it is not certain that any employees will be actually dismissed. If the number of employees drops below the specified minimum, redundant workers are reinstated, and if necessary, new employees are hired.

A company must pay a redundancy payment to any employee who is actually fired, and mass redundancy negatively affects the reputation of a firm on the labour market in the future. Accordingly, a player who wants to reduce employment may specify a maximum number of employees that is higher than the minimum number considered necessary to adjust the number of employees to the minimum that

can be attained without forced redundancies. Another reason to hold on to temporarily redundant personnel is the higher productivity of experienced employees. However, because the number of employee attributes is restricted, individual productivity indices are not maintained, so it is not possible to selectively fire the least productive workers.

Though the prime rationale behind this model was the desire to construct a realistic model with a distinctly European flavour, it also shows the cost differences between a *chase* and a *level* policy in production planning (Vollmann, Berry and Whybark 1992).

Strikes are not implemented in Infogame, though they may have a large impact on company affairs, and are a regular feature in European management games (Van Schaik 1988). First, the duration of a strike is normally much shorter than a quarter, so the outcome of a strike should be decided during a round, which is only possible in a real-time game. Second, whereas it is easy to model the stakes of an employer in strike negotiations, it is very difficult to model the motivations of the opposing labour officials. Consequently, human players acting as labour officials would have to rely on role-playing.

The labour model is described in two stages. First, states and relations between states are described in a global model. A literature search revealed that a similar model was described by Holt (1969). Next, some features of the operational model, that implements the global model by sequential simulation, are described. The global model can also be solved by parallel simulation or by solving a system of simultaneous equations, but in the Infogame context, sequential simulation is more appropriate.

4.4.8.2 Global model description

In the global model, persons enter the labour force as unemployed workers. All unemployed workers and some employed workers are attracted to new job opportunities. The number of persons outside the labour force is not counted; the number of persons entering or leaving the labour force is determined only by the number of employed and unemployed workers. The model is described in Fig 4.6.

All constants and parameters in the model are determined by the game administrator.

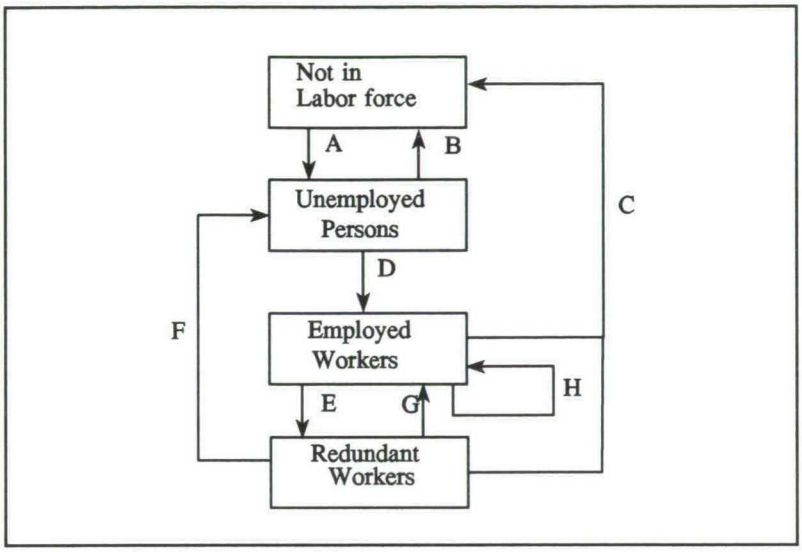


Fig. 4.6: Global labour model

The following assumptions are made:

a) Persons joining the labour force start unemployed, but they may be employed immediately. Consequently, the operational model should compute the additions to the labour force before the hiring of new workers. At the start of the first quarter, the labour force contains only unemployed persons.

b) Workers may be hired at the start of each month. A newly hired worker immediately starts working for his new employer.

c) An employer who wants to fire a worker gives notice at the start of the quarter. Unless the worker departs for another employer or is reinstated, he is actually fired at the start of the third month of the quarter.

d) Both employed and unemployed persons can leave the labour force at any moment. Employed workers do not go into voluntary unemployment.

Transitions in the model (see the letters in Fig 4.6) are:

A,B The number of persons joining or leaving the labour force is determined by the total number of additional employees wanted by employers, the number of unemployed workers, and the total number of employed and unemployed persons currently in the labour force.

C An employee leaves the labour force.

D,G,H An employer can reinstate workers who were given notice and he can hire unemployed persons, workers given notice by other employers, and employees of other firms. The factors determining the employer's success on the labour market will be detailed in the operational model. An employer tries to hire so many new workers that the specified minimum number is attained. Accordingly, he also hires replacements for departing employees.

E) Workers are given notice at the start of the quarter if the actual number of employees is higher than the maximum number specified. However, they hold their jobs during the first two months of the quarter, unless they leave the labour force or find a job with another employer.

F) Redundant workers who do not leave the labour force, are not hired by another employer and are not reinstated by their own employer become unemployed at the end of the quarter.

4.4.8.3 Operational model description

The labour market is simulated in the seven steps represented in Fig 4.7.

1) This step models transition C from the model in Fig 4.6. It is implemented as a loop that ranges over all persons.

2) In this step, the data determining the expansion or contraction of the labour force are collected, and for each firm, as well as for unemployment, a satisfaction index is computed from the relative salary level and the number of person that have actually been fired in the preceding periods. This step is executed as a single loop ranging over all firms.

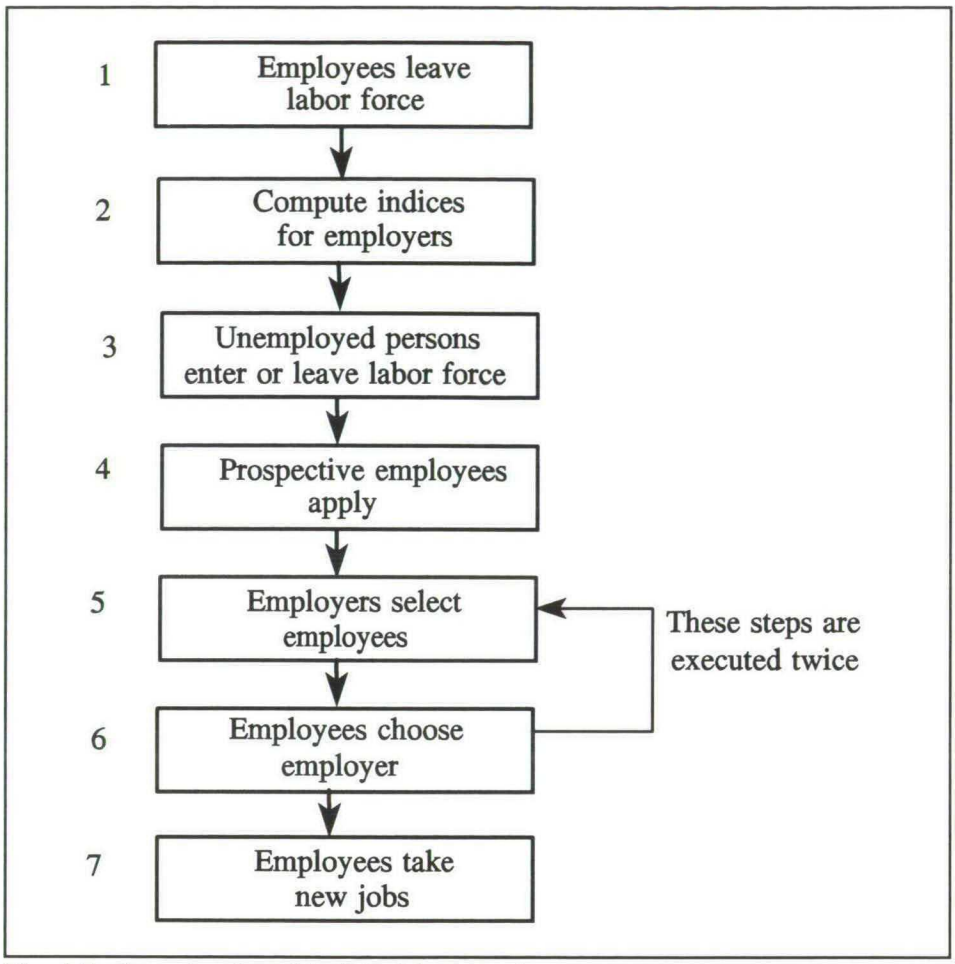


Fig. 4.7: Operational model of the labour market

3) This step models transitions A and B from Fig. 4.6. It is implemented as a loop that ranges over all persons. Accordingly, steps 1,2 and 3 cannot be merged.

4) The persons who will apply for new jobs are drawn. Redundant employees who are still employed will always apply for a reinstatement. An unemployed person or a worker employed by another employer will apply for a job if the satisfaction index of the prospective employer is higher than the satisfaction index of his present employment. A random number models the phenomenon that, at the same time, some workers switch from A to B,

and others switch from B to A. This step consists of an outer loop ranging over all persons and an inner loop ranging over all firms.

5) An employer who has more vacancies than applications, offers a job to every applicant, otherwise one applicant is selected for every vacancy. This is a cautious policy, as it is not certain that all job offers will be accepted. Job offers are made in two stages. First, jobs are offered to redundant employees, next, offers are made to outside applicants. In this step, the outer loop of the double loop ranges over all firms and the inner loop ranges over all persons.

6) A person who has received a single job offer will always accept it. Someone who has received more than one job offer bases his choice between offers on the satisfaction indices. An employee does not always prefer a reinstatement over a move to another job. Contrary to American practice, we do not assume that unemployed workers have a preference for their last employer. Consequently, the identity of the last employer is not stored. When employers compete for a limited number of workers, fewer workers may be hired than needed. This is partially corrected by executing steps (5) and (6) twice, and by filling remaining vacancies in subsequent months. Nevertheless, most workers will be hired at the start of a quarter. As in step 4, the outer loop of the double loop ranges over all persons and the inner loop ranges over all firms. Consequently, steps 4,5, and 6 cannot be merged without changing the model.

7) Acceptance of a job offer triggers a state change. Every person who is hired takes his new job without delay. This step is implemented by executing events initiated by the preceding steps.

4.4.8.4 Labour as a resource

The unit of labour in Infogame is the fully productive worker. The number of fully productive workers in a firm is equal to the product of the actual number of workers and the productivity index (a number between 0 and 1). The number of employees is computed straightforward from the number in the preceding month by

adding arrivals and subtracting departures. The productivity index is computed each month by increasing the productivity of current workers with a learning factor and setting the productivity of new workers to the starting rate. The formula used guarantees that the value of the productivity index ranges between the starting rate and 1. The productivity index was introduced to reward stable employment. Because of the small number of attributes of each employee, it is only dependent on the length of uninterrupted employment with a firm, not on previous jobs or specific assignments. Jobs are not assigned to specific employees. For example, quality cannot be increased by employing more experienced workers.

If the number of employees increases, new employees can only be employed on new jobs. If machines are idle and materials are available, arrivals may trigger the start of a job. Conversely, departing workers are assumed to finish their jobs first. Similarly, the productivity index is only used for new jobs. Current jobs continue with the assigned workers and are finished at the scheduled times. These assumptions were introduced to simplify the model. With average job times, they have no significant effect on results, but in some extreme cases they may jeopardize the verisimilitude of the simulation model, for example when all employees have been dismissed because of lack of cash. Consequently, it may be expedient to adjust job schedules in future versions. The technical aspect of this problem is discussed in section 4.6.4.

4.4.8.5 Discussion

From a theoretical point of view, valid arguments for implementing a detailed labour model have been advanced in section 4.4.8.1, but experience has shown that Infogame users do not develop elaborate personnel information systems. This may either be caused by a lack of verisimilitude in a labour model that simulates employees without individual characteristics, or by the view that, in the real world, implementation and use of personnel information systems does not belong to the core area of information policy. According to the latter view, the labour model could be simplified, but this should not be done before it is confirmed by extensive field research. In the meantime, the realism of the model should be

increased, at least insofar as it was thwarted for technical reasons such as lack of memory, because even if labour policy is considered to have no strategic implications, Infogame may be used for experiments with functional personnel information systems.

The labour model could also be changed to adapt to changes in real-world labour relations. The contrast between the flexible employment of American companies and the stable employment of European organizations has been withering away since the first draft of Infogame in 1985. For example, a future version of Infogame could incorporate the option to hire temporary personnel from specialized agencies instead of directly hiring employees.

4.5 Interface Design

The interface design was based on the character-oriented WYSIWYG screen that was fashionable in MSDOS applications at the time of design. It has not been converted to a Windows interface because the Windows OS scene has not yet stabilized, because the hardware needed for efficient Windows operation is not generally available, and because we did not think the design of a more alluring interface should have the highest priority. An advantage of the Windows philosophy is that input into Infogame and output from a user program that processes Infogame reports can be shown on the same screen, which is not possible in separate MSDOS applications.

In the present version, three mechanisms for input are used:

Selection: An option is selected from a menu, and the program proceeds with the instructions pertaining to that option.

Set selection: A set of options is selected from a menu, and the next instruction operates on the selected set.

Table filling: A table that can be filled in WYSIWYG fashion is presented on the screen, and the next instruction reads and checks the data entered.

Selection and table filling are present in most software packages, but set selection is less common. A selection can never cause an input error, because only correct options are presented on the screen: a management game is no multiple-choice test. Theoretically, a set selection can cause input errors, because the elements of a set may be incompatible, but such errors do not occur in Infogame.

During table filling, two types of input errors may be made, *clerical errors* and *errors in judgment*. The input checker in a management game should check on clerical errors, such as misspelling the name of a supplier, but it should not necessarily signal errors in judgment, such as setting a price that is too low or too high. Not all errors can, however, be classified definitely. For example, trying to start production without acquiring the necessary resources can be considered a clerical error, because an analysis of input data will accurately predict a zero production rate, but the cause of this error may be a fundamental misconception of the simulated system. On the other hand, misstating a number by a factor 10 is a clerical error by nature, but it is hard to detect for an input checker.

In a realistic simulation, an instruction should be challenged if and only if it would be questioned in a real-world environment. This depends on the structure of the organization and the style of leadership. Under authoritarian leadership, instructions will not be questioned, but their consequences will be circumvented. For example, salesmen who cannot compete in the market will exploit the full gamut of discounts instead of telling their authoritarian marketing manager that list prices are too high. Under democratic leadership, instructions will be questioned, and they will be corrected when in error. Democracy is naturally implemented when a game is played by team, with team members checking decisions. If we consider the input program as a middle manager, he expects authoritarian behaviour from the player or player team: input is checked for syntax and completeness only. This approach was chosen for practical reasons. First, corrections or warnings from the input program cannot be used if decisions are written down on forms and entered afterwards. Second, input checking demands a fairly high amount of intelligence in the input program and we did not think this effort our first priority. Possible changes in the input program are discussed in section 6.3.2.4.

4.6 Program and data design

4.6.1 Programming language and environment

Pascal was chosen as the implementation language for Infogame as the best language for which implementations were available on a wide range of computers. The first version was written in Turbo-Pascal Rev 3 on an PC-XT. Later, we switched to Rev 4 to profit from the increased modularity offered by units. Currently, Rev 6 is employed, but its object-oriented features are not used. Meanwhile, the XT was replaced by a 386, and its power has been exploited to increase processing speed and to simplify the program. The power of the pentium that is currently used has still to be employed. Porting between compiler versions asks for a fair number of changes, but such changes are minor when compared to changes in specifications. For example, when a gaming program is ported from a PC to a network or minicomputer, it is a major task to allow simultaneous input from players.

The ideal language would have persistent objects, but the main characteristics of Pascal, such as its procedurality and the prevalence of memory assignment, are not seen as disadvantages. Limitations in memory space and processing speed were seen as a reminder to keep it simple. The resulting program is quite small: it contains some 4500 program lines or 3400 Pascal statements and unlike many commercial software packages, it has scarcely grown since the first version. In future versions, it may be expedient to use different languages for different program modules. For example, a more visual language may be chosen for the interface program.

4.6.2 Program structure

The Infogame system contains three programs: INFOGAME accepts player inputs, INFOMARK simulates the actual operation of companies and markets, and INFOBANK performs special inputs from the game administrator. Data on the initial

state of the simulated world are contained in two ASCII files, INFOGAME.TXT and INFOGAME.LIS, prepared by the game administrator and read by all three programs. Data on state changes are in a file that is used by INFOMARK and INFOBANK. For each player, data entered in INFOGAME are stored in an intermediate ASCII file that can be changed until INFOMARK is started for the next round. INFOMARK retrieves all input data from this file but does not write to it. INFOMARK writes output to ASCII report files for further processing by players and to intermediate ASCII files that are read by INFOGAME. The relation between files and programs is shown in Fig 4.8.

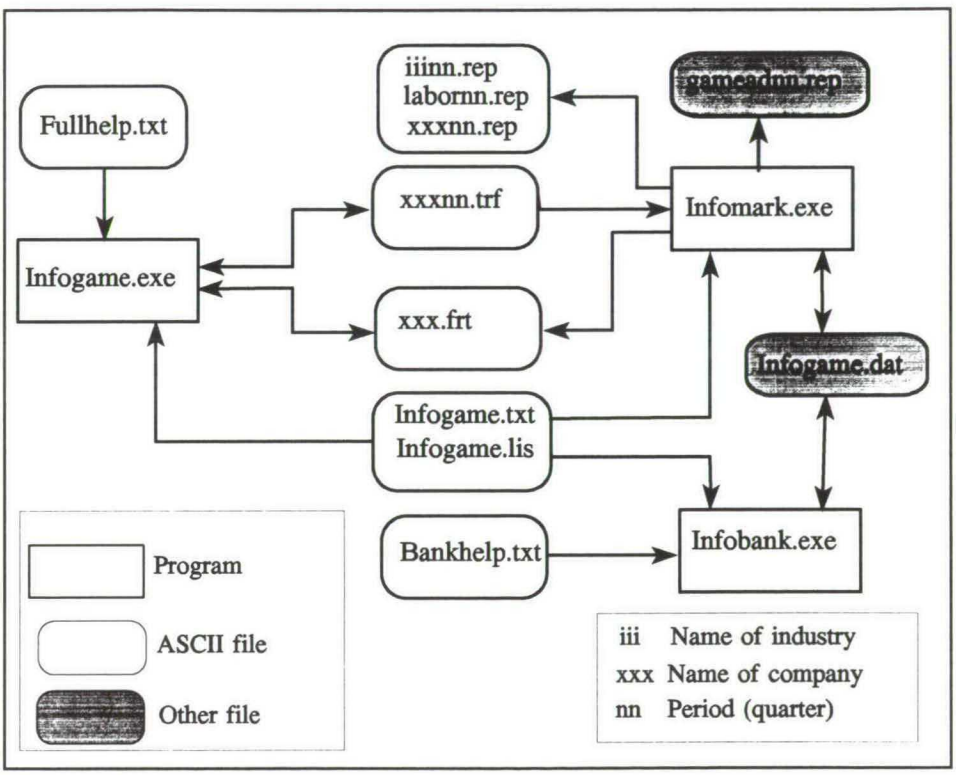


Fig. 4.8: Files and programs

The original reason to split the program into two parts (the functions of INFOBANK were originally performed by INFOMARK) was its size. However, a more important advantage is that the input part can be changed without changing the simulation program as long as it delivers an intermediate file in the same format. For example, in some of the experiments a simple version of INFOGAME with Dutch

texts and default values for many decisions was used. It is also possible to replace INFOGAME with an intelligent program that computes decisions from results in the preceding round. Moreover, the well-defined interface between the two program parts simplifies testing. Accordingly, in chapter 6 a difference is made between changes in the interface program and changes in the simulation program.

4.6.3 Logical data structures

Conceptually, the data structure in Infogame can be represented in the entity-relationship model (Chen 1976; Teory, Yang and Fry 1986). Entities are either concrete entities such as machines, or abstractions such as machine types. Examples of relationships are "use" (a technology uses a type of machine) and "assigned to" (a machine is assigned to a job). A designer of a management game or simulation program has more freedom in conceptual schema design than the designer of an information system in the real world. For example, an earlier version of Infogame incorporated the notion of a *production line*, to which machines and workers were assigned semi-permanently, whereas in the present version, machines and workers are assigned to single jobs. In the design of real-world information systems, such a change can occur only as a result of changes in production organization, and not because it simplifies system design. Another difference is that algorithms, for example scheduling algorithms, are an essential feature of management game programs, whereas the corresponding decisions are treated as external events by the majority of information systems. Accordingly, management games are not suitable for formal methods that derive information systems from conceptual schemata such as RIDL (De Troyer 1993).

In the logical model, a *production order* and a *job* are represented as distinct entities. A production order is created when a product is needed, for example when its stock falls below the reorder level. A job is started when the production factors for a production order are available. However, the same physical data structure is used for both entities. The type of material, which is an entity in the logical model, is represented in the physical model by its quality, which is its only attribute. This conforms to the view that the programmer should be allowed to deviate from the

logical model to improve performance and to reduce program size.

Employees are not represented as entities in the logical model, because it was decided that simulating individual workers with the necessary attributes would take too much memory size and too much programming effort. (see 4.4.8).

The main entities in the conceptual schema are shown in Fig 4.9. For simplicity, attributes are not represented. The relation between consumer orders and production orders only exists in industries that produce to order.

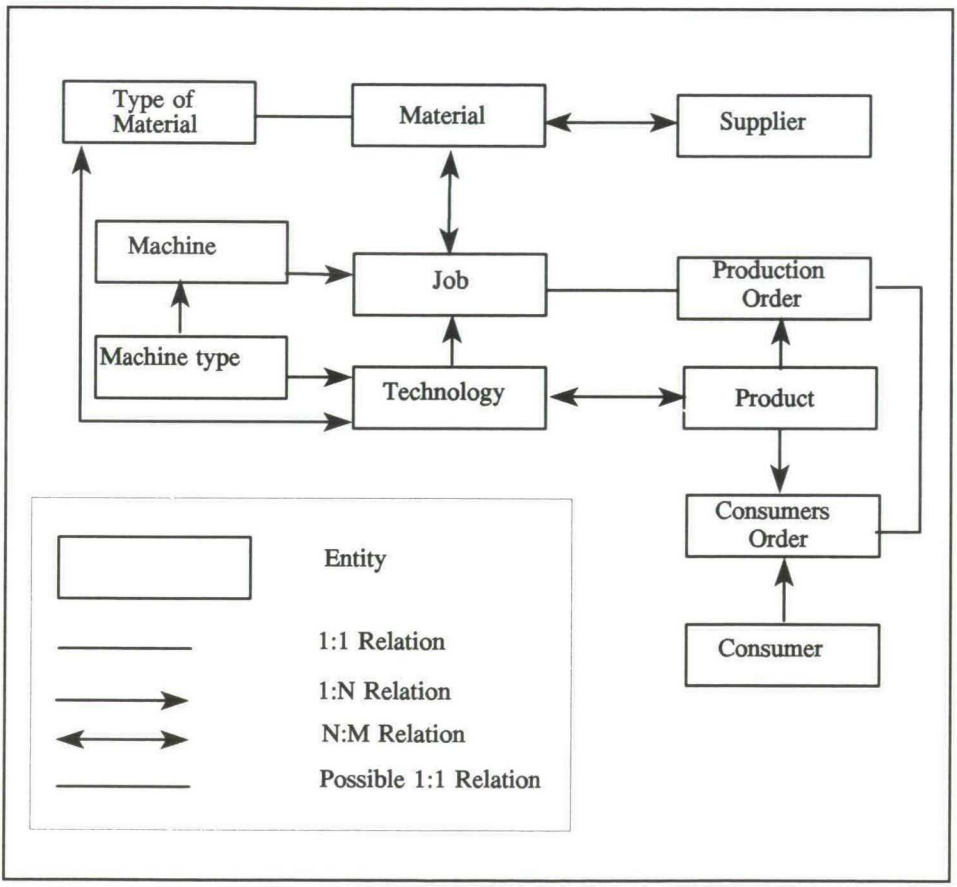


Fig 4.9: E-R diagram of main entities

The logical model that defines the Infogame world can also be used as a logical model for the information system that must be developed for players. However, because there should not be a single "best" information system for use in Infogame, it is not published in the Users Manual. Future extensions should be

designed to increase the number of possible logical models that correspond with the underlying model.

4.6.4 Processes

According to the standard method for event-driven simulation (Kleijnen and Van Groenendaal 1992, p 124), the main loop of the simulation program selects the next event from the event list and executes the procedure pertaining to the type of that event. For example, if the next event denotes the arrival of a consumer, the procedure *consumer arrives* is executed. Most events cause another event in the future by an insertion in the event list. For example, the arrival of a consumer spawns the arrival of the next consumer. The simulation program stops when the event *end of period*, which is inserted at the start of a period, is detected. Apart from procedures that are called by the event scheduler, the simulation program contains procedures that are called directly. In the object-oriented approach, both events with associated procedures and procedures called directly are objects, and all objects are activated in the same manner. In Fig. 4.10, however, relations between procedures are described according to the present implementation.

Coppieters (1990) distinguishes discrete actions or events that occur at a specified moment and continuous actions or *activities*, that proceed during a specified time span. The difference between the two types of actions is easily seen in a military game. If a unit proceeds from A to B, it may be halted because a bridge on its way is blown up before it has been passed, or it may be slowed down because it starts raining. Consequently, the unit's arrival at B cannot be entered in the event list when it departs from A. In Infogame, some activities, such as production of a batch of products, are represented by their start and end, and consequently, they cannot halt prematurely or change speed. Activities may, however, also be represented by data structures, such as the list of production orders that monitors whether production of an order has actually been started. Accordingly, simplifications such as the notion that production batches cannot be interrupted are not really necessary from a technical point of view.

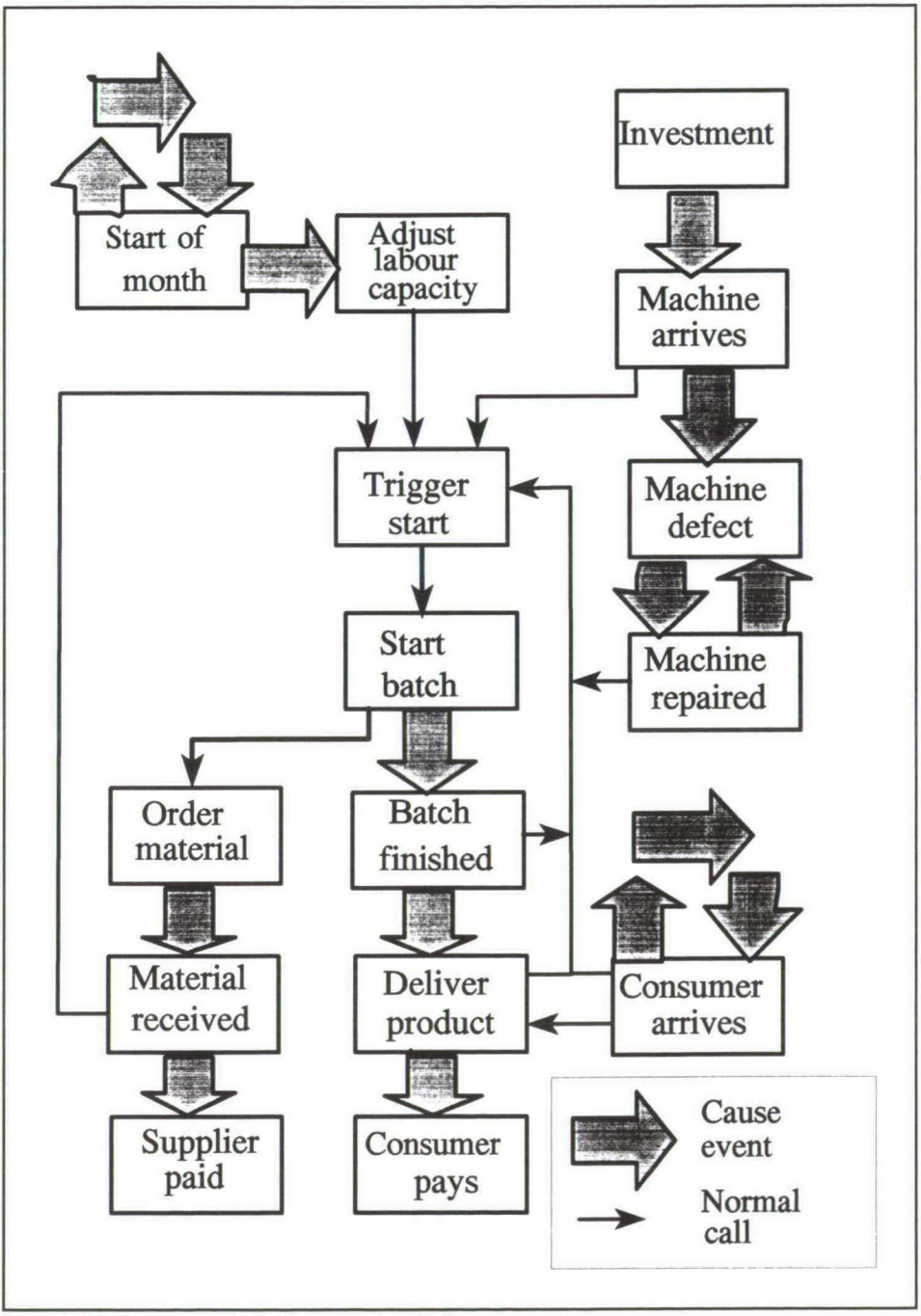


Fig. 4.10: Processes in Infogame simulation program

4.6.5 Physical data structures

In programs that perform elaborate computations from a small set of input data and store all intermediate data in memory, such as simple simulation programs, fast memory access is of primary importance, whereas in programs that are part of an information system, and retrieve and store all data in a DBMS, data integrity is more important. The two types of systems are compared in Fig. 4.11.

Infogame stores data both in memory and on files, because on the one hand, elaborate computations demand fast access to data, and on the other hand, disk backup between playing rounds is necessary to prevent loss of data and to allow restarting a previous round. For files, economy in disk space is more important than fast access. For the memory representation, there is an upper limit to the amount of

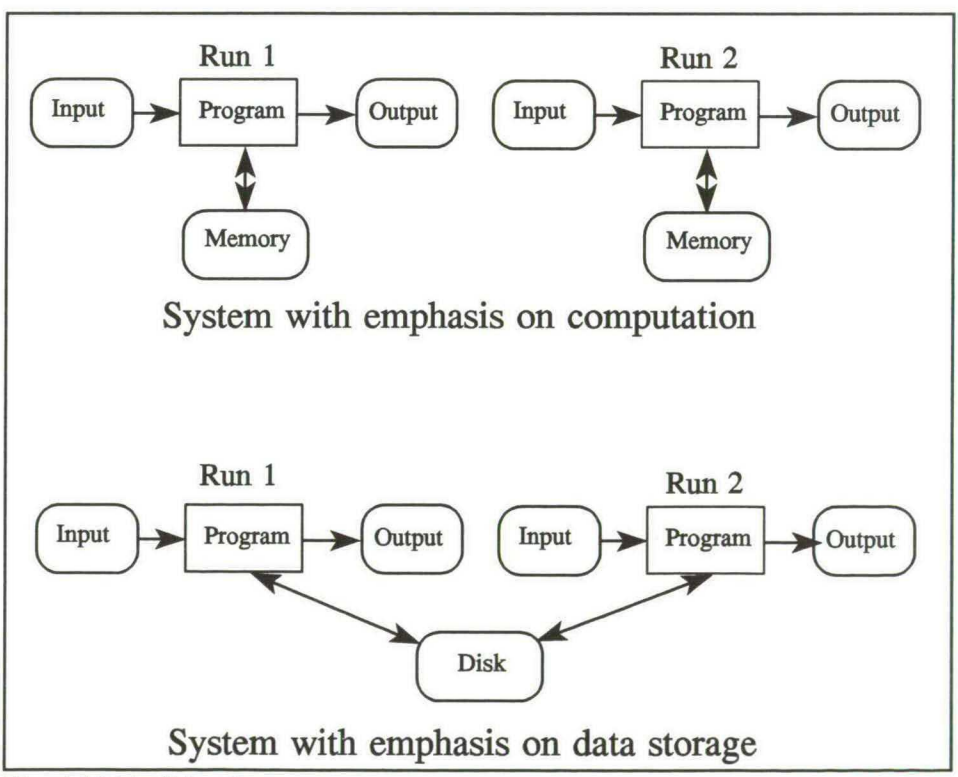


Fig. 4.11 Traditional systems

memory available, but otherwise, speed and simplicity of data access are the prime requirements. This is illustrated in Fig 4.12 .

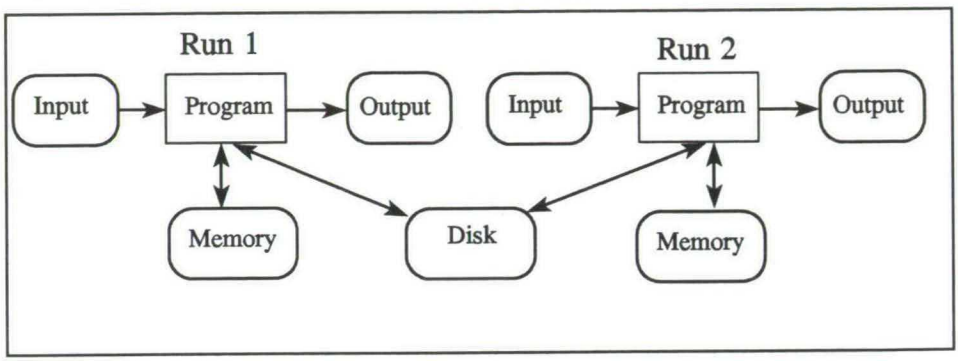


Fig. 4.12 Gaming program

In the design process, the memory representation was defined first. To represent sets of entities, such as machines or products, we had to choose between the linked list and the array (Wirth 1976). Linked lists offer easy insertion and deletion, arrays are more suitable for direct access by index value. The argument that lists in Turbo-Pascal are accessed by pointers and thus use the more abundant heap memory is not valid, because slices of arrays can also be stored in heap memory. An ideal language would support a mechanism for direct addressing by name such as the table in ICON (Griswold and Griswold 1983).

As a rule, we chose the array representation for entities with a constant or limited number of instances. Attribute data for machine types, technologies, and suppliers do not change during the game. They are read from a text file before the start of each round. The introduction of innovation would not necessitate a change in representation, because the resources that are available in a given round can be identified by a variable.

Firms are represented by an array because the number of firms is limited. When a new firm is created, which can occur in any round, it is assigned a permanent number that is also used as an index to the array of firms. It is not necessary to reuse the number when a firm stops operating, because only active firms require memory.

For consumers, the array representation was chosen because the number of consumers remains constant, and growth in consumer expenditure is attained by a

decrease in interarrival time for existing consumers and an increase in the average size of consumer orders. A more sophisticated consumer model, where consumers enter and leave the market, does not require an essentially different data structure.

Employees are also represented by an array. A person who enters the labour force is identified by his index in this array, and memory is reserved to store his attributes. When a person leaves the labour force, his number may be used later for another person who enters the labour market. This was done because the personal history of a worker is not simulated in detail. For example, there are no worker attributes *age* or *professional experience*. Consequently, it was not necessary to use a unique identity number for each person. As discussed in section 4.4.8.5, this simplification could be dropped with the current memory size. An array of sets is used to efficiently store all job applications for each firm. because Turbo-Pascal supports neither one-bit Boolean arrays nor large sets.

Machines, materials, products, and loans are represented as linked lists, but they could also be stored as arrays, because the maximum number of instances of those entities is fairly small. A product is linked to the next product in the firm for the production model, and to the next product in the industry for the marketing model. The representation of production orders and consumer orders as linked lists is essential because the production order list is frequently searched for executable jobs and consumer orders are served on a First In First Out basis.

The event list is structured as a heap, which is a simple method guaranteeing O(n log N) storage time for events (Jones 1986). For each type of event a variant of the event record type is defined. Space allocated to the event list is reused when possible. Once entered in the event list, an event is never deleted before the planned time. Originally, this influenced game design in some details. For example, ending time of a job is never changed once it has been started. In the present version, however, an event can be effectively cancelled because the event list mechanism checks whether the next event must be executed.

At the end of a period, all data needed in the next period are stored in the data file INFOGAME.DAT, and at the start of a period, data are retrieved from that file. Translating all internal data structures into sequential file structures and vice versa requires some 250 lines of logically superfluous Pascal code. Two devices are

used to save file space. Some records are packed in arrays, and other records are saved in separate files. Because the first device suffices for all purposes, the separate files are not discussed here. An additional advantage of storing all data at the end of a round and retrieving it at the start of the next round is that this procedure automatically collects the garbage resulting from the use of linked lists.

Apart from a representation for entities, a representation for relations between entities had to be selected. For all such relations, we had to choose between a single reference involving lengthy computations and double references that required an invariant asserting that the two references stayed equivalent. As the concern for computation speed diminished during program development, we traded speed for security by removing some of the redundant references from the program. In choosing a representation for references we also had to consider that, in Pascal, an element from an array or a data item addressed by a pointer can be retrieved with generic operations [] and ^ respectively, whereas searching a list for a name requires a specialized function. On the other hand, names and indices can be stored in external files, whereas pointers cannot. Accordingly, relations are often represented by names, and in addition to redundant storage and retrieval procedures, the program contains many logically redundant search functions.

4.6.6 Design method

The problem discussed in section 2.3, *how to build a quality information system*, can also be applied to the design of Infogame itself. An important question in this case is whether its design would have profited from the employment of professional programmers. Gremmen (1989) could leave the programming work to professionals because his mathematical model provided a well-defined specification. Coppieters (1990) had to employ programmers because of the sheer size of his assignment. Woltjer (1995) engaged a professional programmer to design a new version of his game after he had demonstrated the educational value of a prototype he developed himself.

In my view, the main advantage of a division of work between designers and programmers springs from the exact description and thorough discussion of the

specifications required. In Infogame, such a discussion would certainly have removed some of its more baroque features, for example in the labour model (see 4.4.8.5). Moreover, the simple programming errors that marred some of our experiments, and that would have been disastrous if Infogame had been played in a professional environment, do not occur in programs written by professionals.

On the other hand, if specifications had been rigorously defined, they would been frozen for a fairly long time, and consequently, it had not been possible to quickly adapt specifications and programs to changing user demands or increasing hardware capabilities. Accordingly, the flexibility in the development of Infogame has benefited from development by a single person.

5 Experiments

5.1 Introduction

In this chapter, the results of experiments with Infogame are described. All experiments have been conducted with students of Tilburg University. Remus (1986) concluded both from a literature search and from an experiment that students could be safely used as surrogates for managers. Moreover, it is not always possible to find other participants. For example, Vennix (1990) employed students in his experiments after unsuccessful attempts to attract policy makers and advisors. In our case, the design of an information system for Infogame takes 100 hours or more, and this makes it nearly impossible to involve actual managers or information systems designers. Another advantage of employing students in our experiments is that they are not yet committed to either the user or the system design side, whereas most practitioners are entrenched in one of the camps. Finally, students could earn a credit by participation in the experiment, and it would be difficult to find a similar incentive for outside participants. Though there are indications that participation in Infogame increased students' awareness of information systems development problems, the educational value of Infogame will not be evaluated in this thesis.

Subjects and experiments are described in section 5.2. Tests of the hypotheses listed in section 2.4 are examined in section 5.3, and results are discussed in section 5.4.

5.2 Subjects and experiments

5.2.1 Overview of the experiments

Tilburg University presently offers, among others, four-year curricula leading to *doctorandus* degrees (roughly equivalent to U.S. masters degrees) in *business economics* and *information systems*. The business economics curriculum (BE) focuses on accounting, marketing, finance and organization, and contains an

introductory course in information systems in the second year. The information systems curriculum (IS) contains similar courses, with more emphasis on information systems, and, in addition, a number of courses in computer science subjects, such as programming and database design. Table 5.1 lists the experiments, the courses involved and the year and curriculum of the students in the courses. In all courses, Infogame was introduced primarily for educational purposes, but at the same time, use of the results for research was planned from the start.

Table 5.1: Use and environment of Infogame experiments

Experiment	Task	Course	Curr-Year
1	Free development	DSS	IS-4
2	Structured development	Introduction to IS	IS-1
3	Specification and design (1)	IS Quality	IS-2,BE-2
4	Use of standard package	Introduction to IS	BE-2
5	Specification and design (2)	DSS+Seminar IS	IS-4+BE-4

Experiments 1 and 5 only delivered qualitative results. A simple statistical analysis was applied to data from experiments 2, 3, and 4. In experiment 1, some students worked alone, but otherwise all subjects worked in teams, so instead of the number of subjects that participated in the experiments, Table 5.2 lists the number of teams from which data were collected. In addition, the number of replications of each experiment with different classes and the average time devoted to Infogame by students in the experiment are given.

Table 5.2: Infogame experiments

Experiment	Semesters	Classes	Teams	Time (hours)	See
1	Fall 90-93	4	20	50	5.2.2
2	Spring 93-95	1	43	100	5.2.3
3	Spring 93	3	71	10	5.2.4
4	Fall 93	1	17	20	5.2.5
5	Fall 94	1	4+6	50+20	5.2.6

5.2.2 Free development

The experiment conducted in the Decision Support Systems course can be labelled as free development. The DSS course is an elective course for information systems students in their final year. It has been taken by 8 to 12 students annually. The main purpose of the course is to teach students to design, implement and use decision support systems. The course starts with a series of simple assignments, mainly variations of problems found in Turban's (1993) textbook, that must be solved with spreadsheets. Consequently, though students are free to choose any programming language or application package for the Infogame assignment, most of them have used a spreadsheet. The Infogame assignment contained three steps:

1 Students, working alone or in teams of two or three, designed and built an information system for the full version of Infogame. No formal development method was prescribed.

2 Students played three or four rounds of Infogame. The game was played one round a week, so students had ample opportunity to use and update their information systems.

3 Students wrote a report on their information system and its use in the game. Both this report and the final program were evaluated by the course lecturer.

Because Infogame was fundamentally changed between replications, the number of player teams in this experiment has been far too small for statistically significant conclusions, but results suggest that players often underrate the amount of data needed for effective decision support, and that huge losses are often caused by simple errors, such as forgetting to order necessary materials, rather than to erroneous strategic decisions. Another recurrent error was the wrong focus. For example, in the 1993 course, one team considered its relative low market share as its main problem, although its information system duly reported that it could not even meet promised delivery times for the few orders it did obtain.

5.2.3 Structured development

Structured development was practised in the introduction to information systems course for first-year students in information systems. In the context of a newly introduced system of *student-centred teaching*, those students, working in teams of three or four, designed and built an information system according to the traditional Life-Cycle model for a simple version of Infogame with a single industry. All students had to follow a linear development method, and intermediate reports were discussed with student-assistants. When the systems were planned to be completed, Infogame was played, and finally teams produced a report, that also contained all previously delivered milestone products. Though the actual number of teams was somewhat larger, data were collected from 12 teams in 1993, 16 teams in 1994, and 15 teams in 1995.

In 1993 and 1994, five or six teams competed in one of four or five "worlds". The first four quarters were played according to a fixed scenario and the resulting data were made available to students at the beginning of the exercise. The game was further simplified by allowing only production for stock. A spreadsheet (Lotus 1-2-3 or Quattro-Pro for Windows) was used for implementation, because this was the only package which was known to all students. Infogame play consisted of playing four additional quarters with some days between sessions to allow for corrections to the information system.

In 1993, information on competing companies was published in a "newspaper". Accordingly, players could use information that was not present in their own information system. In 1994, consumer and labour market data were directly available to players (see 4.3.6).

In 1995, teams competed in two worlds with nine or ten teams each. Companies started from scratch in the first quarter, and all six rounds were played in a single day. Accordingly, information systems could not be adapted during use, but programs could be tested with a set of test data that was available to all teams. The game parameters had been changed to permit back-orders. Because the curriculum now includes Pascal programming in the first year, systems were implemented in Pascal, but building and testing a spreadsheet prototype before programming was

highly recommended.

In all experiments, information systems development was based on the standard waterfall model, and contained the following phases:

1 Preliminary design
2 Global design
3 Detail design
4 Implementation
5 Use

Preliminary design coupled the design of the information system to a general information systems concept, for which the Blumenthal model (Davis 1974) was generally used. In this phase, the information system was usually defined in very abstract terms only. Because we could not determine a suitable metric for the quality of this milestone product, we did not use it in our analysis.

Not all teams made a clear distinction between *global* and *detail* design. When the distinction was made, global and detail design differed both in the topics considered and in the detail of the description. The set of data on global and detail design is the most complete set of data available, and consequently, it will be used to test our hypotheses in sections 5.3.2 and 5.3.3.

Implementation entailed programming in a spreadsheet language or Pascal. For 1993 and 1994, only a minority of the spreadsheets was available, and, moreover, systems were improved during and after actual use. The 1995 data were not yet available when this chapter was written. Consequently, it was not possible to examine the relation between design and implementation quality.

Use of information could be inferred from reports on the decisions taken and the information used for those decisions. These showed that information systems were not always fully used, but data on this topic were not collected in a systematic way. Company profit can be considered the ultimate yardstick of the use of an information system. Because profit data were available, section 5.3.6 relates company profit to design quality.

This experiment delivered the most reliable data for further analysis because

it involved a sufficient number of teams, because students in this experiment invested a considerable amount of effort in designing an information system, and because the courses were adequately supervised by student-assistants.

5.2.4 Specification and design (1)

In the first experiment on specification and design, second-year students in information systems and business economics specified the requirements for an information system that could be used in Infogame. 71 teams of 2 to 4 students produced a specification. Next, a program purporting to embody those specifications was written by the game administrator. According to the original plan, about half of the programs would deliberately diverge from the specification, either by the addition of unspecified features or by the omission of specifications, but for educational reasons, all specifications were implemented more or less accurately. Subsequently, Infogame was played in "worlds" of four to six teams for four quarters according to a fixed scenario, and the student teams used "their" information system to analyze results and to play the fifth quarter. Finally, student teams evaluated the results of the fifth quarter and rated the information system.

In this experiment, a large number of programs had to be written, often from inconsistent or incomplete specifications. Because it was decided beforehand that the specifications should not be reviewed, slight flaws in specifications, such as an omission of the frequency of reports, which stemmed from oversight rather than lack of insight, could have dire consequences.

The results from this experiment may not be quite reliable because the grades assigned to the quality of information systems sometimes represented a view on the game instead of a view on the simulated environment. For example, low grades for reliability were given because the network used did not adequately protect data. Moreover, it is clear that one period of play is too short for conclusions on the relation between information systems quality and game results.

5.2.5 Use of a standard package

20 teams of 3 or 4 business economics students in the second year course "Principles of information systems" participated in this experiment. The course itself is compulsory for business economics, but participation in the experiment was optional. The experiment consisted of the following steps.

1 Student teams specified a list of requirements for an information system.
2 Infogame was played for four quarters according to a fixed scenario. Teams were divided into four worlds with five companies each.
3 A manual describing a standard package written by the game administrator was handed to students. It was important that students should not read this manual before they handed in their specifications, lest they should be influenced by it.
4 Students teams analyzed company results of the first four quarters by selecting up to ten reports, with up to ten variables per report, from the standard package.
5 Student teams played the game during four more rounds. After each round the selected reports and variables could be changed.
6 Student teams evaluated their play and the information system built with the standard package.

5.2.6 Specification and design (2)

The second experiment on specification and design involved students from two different courses. Six teams of three or four BE students, participating in a seminar on information systems, played Infogame, and three teams of two or three BI students from the DSS course designed an information system according to the specification of one or two BE teams. Moreover, one BI student designed an "application package" for investment selection. The cooperation between teams is described in Table 5.3.

Discussions between users and designers started with misunderstandings

about expectations. For example, some user teams were unsure what type of specification was needed by the DSS teams. In a later stage, a user team failed to mention that it only produced on order, whereas the DSS-team put much effort into designing an information system for control of stocks of finished products. Personal affinities sometime decisively influenced cooperation between teams[19], and failures in communication, for which E-Mail had to be used, seriously hindered systems development and use.

Table 5.3 Designers and users

Playing team	DSS team	Specification	Use	Profit M
1	1	Given	Low	4,3
2	2	Given	Low	1,6
3	2	Designed own system		13,6
4	3	Given and updated	High	-6,1
5	1	Updated	Medium	4,6
6	3	Not given	None	-4,7

Although the team that designed its own DSS was the most successful, it stated in its report it did not consider DSS design a proper task for managers. The loss of the firm that had the best cooperation with its designers stemmed from an initial investment decision.

[19] For example, a user team mentioned that an all-male DSS team became far more willing to answer questions when they found out that the user team included two girls.

5.3 Testing hypotheses

5.3.1 Hypotheses

In section 2.4, the following hypotheses were proposed:

Hypothesis 1: In an information system, the quality of a milestone product depends on the quality of the preceding milestone product.

Hypothesis 2: In an information system, the quality of a milestone product depends on its correspondence to the preceding milestone product.

Hypothesis 3: User satisfaction with an information system is higher if there is a closer correspondence between specification and implementation.

Hypothesis 4: Users who prepare clear preliminary specifications will have more success in implementing and using a standard package than users who do not.

Hypothesis 5: Organization success, as measured by company profit, is positively correlated with information system quality.

As explained in section 5.2, only experiments 2, 3, and 4 produced quantitative results. The experiments where the hypotheses were tested are listed in Table 5.4. As related in section 5.2.3, the tests of hypotheses 1 and 2 involved global design and detail design only.

Table 5.4: Experiments and hypotheses

Hypothesis	Experiments	Description
1	2	5.3.2
2	2	5.3.3
3	3	5.3.4
4	4	5.3.5
5	2,3,4	5.3.6

5.3.2 Test of hypothesis 1

To test hypotheses in experiment 2, we had to develop a measure of quality that could be applied to different phases of the development cycle. This quality measure is derived from Bedell's (1985) *ESA index*, which is equivalent to the *completeness* characteristic used in some other scoring models (Bailey and Pearson 1983; Delen and Rijsenbrij 1990a). However, from the three aspects covered by the ESA index, *functional appropriateness, technical appropriateness,* and *cost-effectiveness*, the latter two can be defined for completed systems only. Instead of using Bedell's coarse-grained yardstick, which only distinguishes highly effective, moderately effective, ineffective and useless information systems, we graded information systems on a 0 to 10 scale. Characteristics and weights were derived from the notion that a satisfactory information system should cover both planning and feedback, and that feedback should be available for all major functions in the simulated company, i.e. marketing, finance, and production. The weights used in the scoring model are given in Table 5.5. Each characteristic consists of 3 to 8 components, and the quality of global design and detail design was subjectively assessed by the author. The assessment was based on precision as well as completeness. For example, the specification that the information system should list sales data by product is more precise than the specification that it should list sales data, and hence the former specification is considered better. For 1993, *competition* and *labour market* were not included in the score, because information on consumer and labour markets could not be derived from players' information systems.

Table 5.5: Characteristics and weights

Characteristic	Weight 1993	Weight 1994-95
Plans	0.25	0.2
Sales	0.25	0.2
Competition		0.1
Finance	0.125	0.1
Product flow	0.125	0.1
Product state	0.125	0.1
Resource state	0.125	0.1
Labour market		0.1

A preliminary analysis of data shown by Fig. 5.1[20] established that the quality of global and detail design was related, and also that the quality of detail design was significantly higher than the quality of global design.

Accordingly, hypothesis 1 was replaced by hypotheses 1A and 1B:

Hypothesis 1A: In an information system, the quality of a detail design depends on the quality of global design.

Hypothesis 1B: In an information system, the quality of detail design is higher than the quality of global design.

For the test, we used 43 instances for which the quality of global and detail design could be evaluated. Data from 1993 and 1995 were converted to yield the same average as the 1994 data. As shown in Table 5.6, both hypotheses are confirmed at the 0.05 level. The implications of these results are considered in section 5.4.1.

[20] Statistics and graphs have been produced with Quattro-Pro for Windows Rev. 5.0.

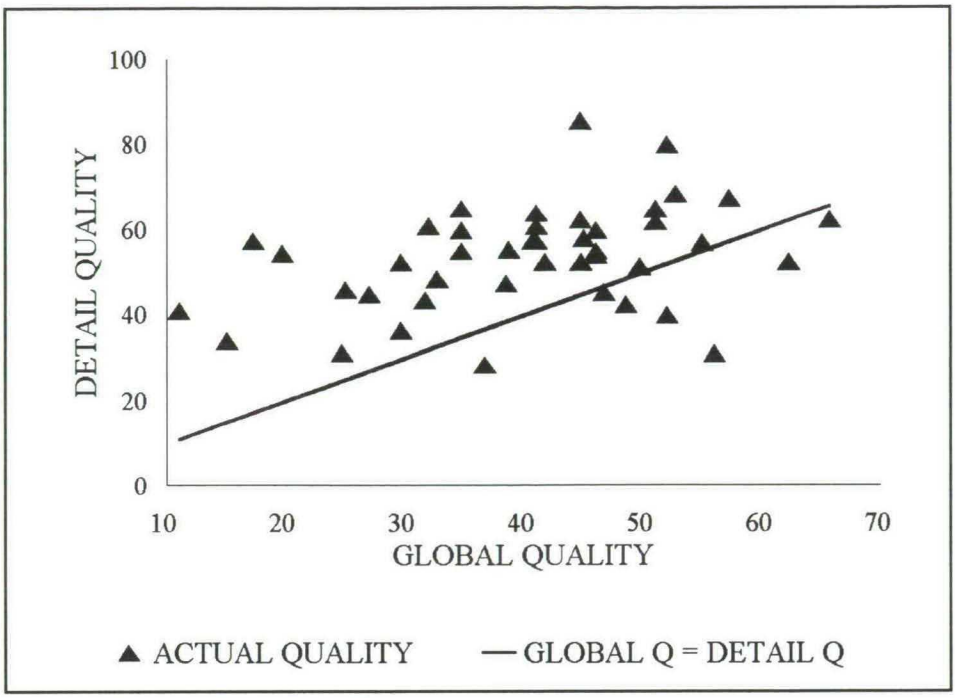


Fig 5.1: Global and Detail Quality

Table 5.6: Global and detail design

	Coefficients	Standard Error	t Statistic	P-value
Intercept	39.16855	5.935237	6.599324	5.45E-08
x1	0.359602	0.140577	2.55805	0.014223

5.3.3 Test of hypothesis 2

For each of the 43 instances mentioned in section 5.3.2, we computed the relation between the number of changes from global to detail design and the quality of detail design. The number of changes is counted by summing the features in the global design omitted in the detail design and the new features in the detail design. The result is given in Table 5.7.

Table 5.7: Changes and quality of detail design

	Coefficients	Standard Error	t Statistic	P-value
x1	1.104040732	0.373330353	2.95727557	0.005076719

The results in Table 5.7 not only refute hypothesis 2, which postulates a negative correlation between changes and quality, but they show that detail quality is positively correlated to the number of changes. This can be explained by the increase in quality from global design to detail design, which apparently dominates other differences between global and detail design.

Subsequently, we tried to determine *what* changes were made between global design and detail design. To this end, in Table 5.8 we counted how often the most important variables were specified in global and detail designs.

Table 5.8: Variables in global and detail design

Variable	Detail design		Global design	
	Number	Rank	Number	Rank
Number of employees	38	1	37	1
Total sales	36	2	26	3
Cash/Loans	32	3	24	4
Machine capacity	32	4	35	2
Stock of materials	31	5	16	9
Loans	25	6	15	10
Total profit	25	7	18	7
Stock of finished products	25	8	18	6
Total costs	23	9	9	17
Cost price	22	10	5	26
Production	16	17	18	8
Market Demand	5	38	18	5

Table 5.8 shows that the same variables are specified in global and detail designs, but that they were more often specified in detail designs. This suggests that detail designs were often used to fill gaps in the global design.

The difference between global and detail design can be explored in more detail by examining the number of times a variable is mentioned in one of the designs and omitted from the other design by the same team. Table 5.9 lists the variables most involved in a change.

Typically, variables only mentioned in the global design are of a general nature, whereas detail design involves more specific variables, including cost price and depreciation, which are in the business domain rather than the information systems domain. The implications of this result are discussed in section 5.4.2.

Table 5.9: Differences between global and detail design:

Global design		Detail design	
Variable	Times	Variable	Times
Market Demand	15	Cost price	18
Labour market	9	Total costs	18
Production	8	Stock of materials	17
Labour productivity	7	Interest payments/receipts	15
Machine capacity	6	Depreciation/Book values	14

5.3.4 Test of hypothesis 3

In experiment 3, the relation between specification and implementation was not controlled, as originally intended, and it showed insufficient variation to be used as an independent variable. Hence hypothesis 3 could not be tested, and we replaced it by hypothesis 3A to test the influence of a different factor on perceived quality.

Hypothesis 3A: User satisfaction with an information system is positively correlated to the completeness of the implementation.

Because we assume that implementation closely follows specification, completeness of the implementation should imply completeness of specifications. When it is also assumed that all specified items serve some purpose, program size, measured by number of statements, can be considered a fair measure for implementation completeness. In essence, hypothesis 3A is the converse of hypothesis 3, because it states that perceived quality depends on an intrinsic characteristic of the implementation, whereas hypothesis 3 states that it depends on the relation between specification and implementation. For the 52 teams that delivered a quantitative quality index, no significant relation between program size and quality index was found. The results are shown in Table 5.10 and Fig. 5.2.

Table 5.10: Program size and quality

	Coefficients	Standard Error	t Statistic	P-value
x1	0.005168	0.003335	1.549470	0.127333

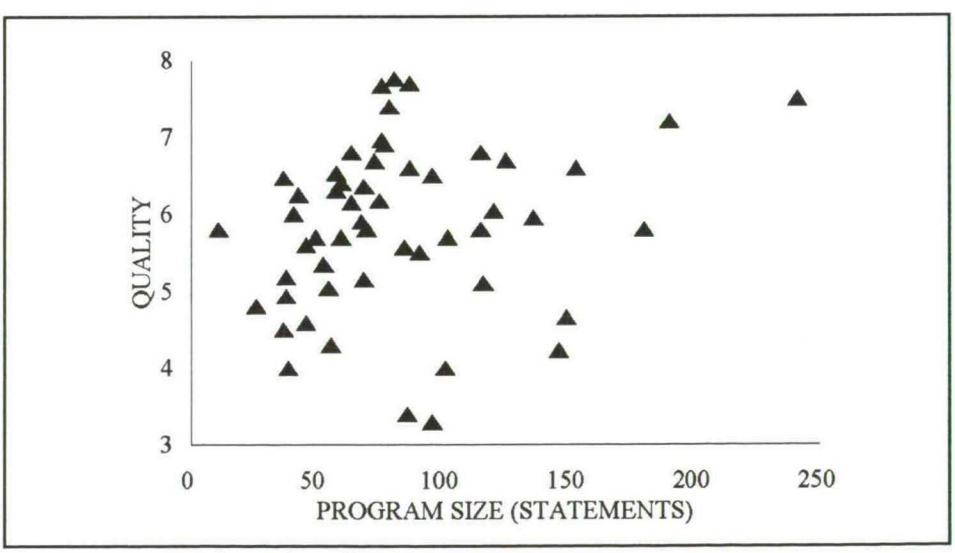


Fig 5.2: Program size and quality

5.3.5 Test of hypothesis 4

In experiment 4, we analyzed the list of requirements, the variables reported, and the results of 17 teams. The view that preliminary specifications are important for successful implementation of a standard package is embodied in hypothesis 4:

Hypothesis 4: Users who prepare clear preliminary specifications will have more success in implementing and using a standard package than users who do not.

Instead of directly testing this hypothesis, we first try to determine whether preliminary specifications are really used in the implementation of a standard package. To this end two new hypotheses are formulated:

Hypothesis 4A: The number of times a variable is specified in the standard program depends on the number of times it is included in the preliminary specification.

Hypothesis 4B: The number of variables specified in the standard program depends on the number of variables included in the preliminary specification of the same team.

Hypothesis 4A states that, in the standard program, players ask for the variables that were included in the preliminary specifications. Hypothesis 4B states that users with longer preliminary specifications ask for more items from the standard program. Neither hypothesis was supported by the data in Table 5.11.

Table 5.11: Specification, use and profit

		Coefficients	Standard Error	t Statistic	P-value
4A	x1	0.222522	0.129062	1.724140	0.0908602
4B	x1	-0.332445	0.256395	-1.296608	0.213154

5.3.6 Test of hypothesis 5

The relation between information systems quality and profit has been examined before in a number of studies (see 3.3.4). First, Hypothesis 5 was tested with data from experiment 2. Because data from 1995 were not yet available, only the 28 data items from 1993 and 1994 were used. The independent variable is the quality index of the detail design, described in section 5.3.2, and the dependent variable is the cumulative profit in the four quarters for which players made decisions. The results are given in Table 5.12, and data are plotted in Fig. 5.3. The test showed no significant relation between quality of detail design and profit.

Table 5.12 Quality of detail design and Profit

	Coefficients	Standard Error	t Statistic	P-value
x1	22.3861	66.30364	0.33763	0.738253

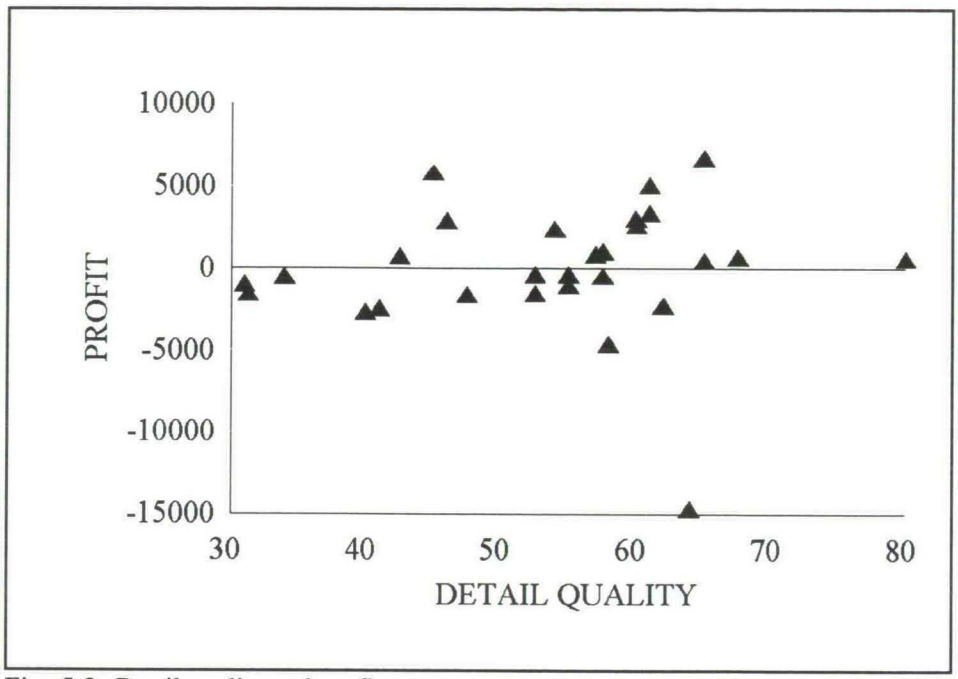


Fig. 5.3: Detail quality and profit

Second, data from experiment 3 were used. Both quality and program size were used as independent variables, and company profit in the quarter for which players made a decision was used as the independent variable. For the sample of 71 teams, no significant relation between program size and company profit was found, but for the 52 firms for which a quality index was available, the relation between quality index and profit was just significant at the 0.05 level.

Table 5.13: Quality, Program size and Profit

X	Y	Coefficients	Standard Error	t Statistic	P-value
Quality	Profit	137357.6	67679.42	2.029532	0.047536
Program size	Profit	1627.156	1589.499	1.023691	0.309456

Third, we studied data from experiment 4. In this case, the independent variable, information systems quality, was defined by the correspondence between

variables specified and variables used, and the dependent variable was cumulative profit for the last four quarters. Table 5.14 shows that no significant correlation has been found.

Table 5.14: Specification, Use and Profit

	Coefficients	Standard Error	t Statistic	P-value
x1	93.4358	115.810	0.806796	0.431613

5.4 Discussion

5.4.1 Quantitative results

The main quantitative result of our experiments is shown in section 5.3.2, which demonstrates that milestone quality increases from global design to detail design, and that quality in the two phases is related. On the one hand, these results seem trivial, because it is quite understandable that information systems designs improve during the development cycle, especially if designs are made by undergraduate students. On the other hand, just because results conform to conventional wisdom, they demonstrate that our method can be used for further research into the determinants of design quality. Two obvious candidates for research are the use of prototyping and the application of formal methods.

The test of hypothesis 2 in section 5.3.3 shows that high quality in the detail design phase is often attained by improvement during the life cycle. Further experiments are needed to determine whether this holds under different circumstances. For example, it is quite possible that teams that are forced to spend more time on global design will eventually produce better detail designs than teams that make an early start with detail design.

The results reported in section 5.3.4 show no significant correlation between program size and quality index. This implies that users are not impressed by large programs, that are based on lengthy specifications. The design of further tests on this

topic is discussed in section 5.4.3.

The test of hypothesis 4 in section 5.3.5 gives no support for the notion that implementation of standard packages profits from preliminary planning. Because this is counterintuitive, further research on this topic is needed.

With the exception of a barely significant relation between perceived quality and profits, which may be attributed to the fact that players give a higher score if they meet success in a game, the data in section 5.3.6 show no support for hypothesis 5, which implies there is no evidence that information systems quality contributes to the profit of the firm. Because this result disagrees with the well-known fact that rational organizations spend huge amounts on information systems, some possible reasons for this effect should be given.

First, the use of profit as a yardstick for organization success must be examined critically. Instead of profit, other measures, such as the balanced scorecard (Kaplan and Norton 1992) could be used. If the primary goal of companies is profit satisficing rather than profit maximizing, successful firms will be characterized by other attributes, such as outstanding products, rather than superior financial performance, just as nonprofit organizations are distinguished by the quality of services, and consider costs and income as restrictions. In an experiment, teams could decide to reduce management efforts when results are satisfactory. On the other hand, failure can always be defined in financial terms, because an organization can only survive if it is financed by sales or subsidies. Accordingly, a correlation between information systems quality and profit, may only be found in firms with unsatisfactory profits, but in a laboratory setting, the number of such firms is usually too small for statistically significant results. Consequently, this topic is further discussed in section 5.4.2.

The study of the relation between information systems quality and organization performance may also be hampered by environmental factors. For example, the really important information may not be available to players, all players may have designed satisfactory information systems, information systems may have been designed, but not used, or the players, being first or second year students, may just be unable to make adequate business decisions. On the other hand, even if a correlation between information systems quality and profit had been found, there

would have been reasons to doubt its external validity, because the experimental environment may not adequately reflect the complexities of real organizations.

5.4.2 Qualitative results

The increase in quality during the system development life cycle shown in sections 5.3.2 and 5.3.3 suggests that the value of the waterfall model primarily originates from its contribution to the learning process of designers. Contrary to the canonical view of the waterfall model, which implies that business knowledge is primarily needed in the specification phase, the transition from global design to detail design required detailed business knowledge. Further research on this topic calls for experiments with a strict division between a business team that specifies and uses the information system and an information systems team that designs the information system. For example, the importance of business knowledge in the design phase can be determined by comparing information systems designed by teams with different backgrounds.

In experiment 2, we analyzed the results of all 1994 teams with considerable losses. From the eight teams considered, one did not finish its information system in time, and could not use it for decisions. Three reams had trouble with tuning production and marketing. For example, the team with the highest loss consistently operated far below capacity because of an incorrect relation between reorder level and order quantity. Four teams made elementary mistakes because they misunderstood the environment. Such errors occur in the real world as well, especially when a company enters a foreign country or industry. The errors made by these four teams were:

1 Machines or materials that had been ordered could not be paid. Consequently, planned production was not possible. This is an error in *financial planning*.

2 There was no correct match between existing or newly ordered machines and the specified technology. This is an error in *production planning*.

3 The materials needed for production with the specified technology had not been ordered. This is again an error in production planning.

All such errors can be noticed beforehand. Indeed, correct matching of machines, technologies, materials, and financial resources is an appropriate subject for an expert system like the DEC system which guarantees the completeness of configurations. Moreover, a well-designed information system would have signalled them in time to take corrective action. This did not always happen, and some teams never found the cause of their troubles. In the real world, elementary problems are usually signalled at the operating level. However, this will not immediately remove their causes, or prevent similar problems in the future. Moreover, the growth of production automation will increase the importance of error control within the information system.

From these cases, we conclude that, in several instances, failure of a firm can be explained by an inadequate information system. Continued research on such cases may lead to a better understanding of the relation between information systems quality and organization success.

In experiment 4, an unanticipated observation was that incorrect numbers, that stemmed from more or less random programming errors, were believed at face value, although the error could have been detected by simple computations. Though research on this phenomenon is certainly warranted, it can also be executed in a simple laboratory setting or in a conventional management game, where errors can be deliberately seeded into reports.

5.4.3 Design of experiments

Two experiments described in section 5.2 will not be continued. For educational reasons, free development (see 5.2.2) is replaced by design to the specification of outside users (see 5.2.6). From the research point of view, this is no great loss, because the number of variables in free development projects is too large to produce significant results. Design to specifications by a single designer (see 5.2.4) will not be repeated because of the unsatisfactory relation between effort and

results. Two alternatives are the use of a standard package (see 5.2.5), and the employment of students as information systems designers (see 5.2.6).

The experiment with structured design (see 5.2.3) will be continued more or less in the same form. For research purposes, apart from data on global and detail design, more data will be collected on preliminary design, implementation and use. We think the use of Pascal will encourage more systematic programming, and less emphasis on the bells and whistles that come with spreadsheets such as Quattro-Pro for Windows. This will also improve comparisons between implementations.

For research purposes, we would prefer to perform this experiment with more advanced students, but from an educational point of view, the present setup offers an unique opportunity for first-year students to engage in a medium-scale information systems development project.

The experiments with a standard programming package should also be repeated. Advantages of this experiment are that data on the choices of players are automatically collected, and that this exercise asks relatively little time from players.

The experiment with separate users and designers showed that Infogame is quite suitable for exercises in cooperation between actors with different roles, which are often thought to be the preserve of social simulation (Van der Meer and Roodink 1991). To get meaningful results, data must be collected on the independent variables, i.e. the quality of the specifications, the quality of the information system design process and the quality of cooperation between managers and designers, and the dependent variable, i.e. the quality of the information system. An experiment along these lines has been started with six user teams and seven design teams.

6 Conclusions and further research

6.1 Introduction

To evaluate the research described in this thesis, two topics are investigated. In section 6.2, experimental research with Infogame is compared to other research strategies. In section 6.3, the extension of the present use of Infogame to other subdisciplines of information systems by changes in Infogame is discussed. Section 6.4 provides an overall evaluation of the research project.

Though we think Infogame is a worthwhile educational tool, we have not specifically studied its value as such by comparing results of Infogame users and nonusers. For two reasons we think its use in education is intimately tied to its use as a research tool. First, in our view, education offers the best opportunity to study highly motivated information systems designers. In this respect, management games provide a challenge that is absent from other types of laboratory research. Managers have been enthusiastic players of management games, whereas it is doubtful that they will engage in simpler tests. Moreover, because of rapid technical progress, information systems education is not limited to regular students. For example, the use of programming languages can be studied during 4GL courses for experienced COBOL programmers.

Second, in our opinion, a new game that delivers no relevant research data can have no added value for education. If observing player behaviour in a new game provides no additional knowledge, the lessons from the game can as well be taught straightforwardly from a textbook. As an example, suppose that management game players determine advertising budgets by looking to competitors' budgets rather than by applying an analytical model. Now, either we have a research result, which tells that, under certain circumstances, advertising budgets will be set by comparison with competitors, or we deny the validity of the experiment because we know from field research that real-world marketing managers actually use analytical models. In the latter case, the management game teaches the wrong lesson.

However, the validity of research results depends on the similarity of

players and represented actors. For example, if a game that simulates an African Village market (Greenblat 1987) is played by European students, the results will contribute no knowledge on conditions in Africa. Moreover, the educational value of a management game is not dependent on the novelty of the research results, but only on their validity. When a game has been used for some time, its potential research results will be generally known in the research community.

6.2 Research strategies

6.2.1 Introduction

Research with Infogame can be replaced or supplemented by other types of research. In this section, four competing research strategies are discussed, *traditional laboratory research*, *theoretical research*, *engineering research*, and *field research*.

6.2.2 Traditional laboratory research

Traditional laboratory research involves simple tasks that can be completed in a few hours. As shown in chapter 2, in information systems development, this type of research can only resolve problems pertaining to single phases of the development cycle, such as the relative quality of programming languages or other design tools. However, it is even arguable whether answers to such problems can be given in isolation. For example, when an experiment with small example problems shows that programming in language A is more efficient than programming in language B, the result is not necessarily valid for programming large-scale information systems. Infogame research supplements, rather than replaces traditional laboratory research by offering an opportunity to corroborate small-scale laboratory research results with lower cost than full-scale field research.

The most promising field for all types of laboratory research in information systems is the evaluation of tools and languages. Considering the huge sums spent to design and implement programming languages, compilers and CASE tools, it is

remarkable that so little empirical research is done on their use. Probable, the situation is not as bleak as could be inferred from the published literature, because conclusions drawn implicitly from student and practitioner use may have been published in the "speculative" literature.

6.2.3 Theoretical research

In the field of information systems development, new theories and applications of existing theories to new fields continually emerge. According to the present practice in the field, such theories can be published before empirical work has been done on the subject. Some researchers even depart for other fields when a research area is subjected to the harsh rules of mathematical proof and statistical verification.

Whereas theoretical research of a speculative nature often precedes empirical research, the other blend of theoretical research, theorem proof research, is usually executed at the same time. The two types of research are used for quite different purposes. Theorem proof research can determine the equivalence of representations, and experimental research can be used to test human reactions to representations. Theorem proof research is especially appropriate for the design of the data processing part of a system that can be described by a mathematical model, for example in the design of a fully automated warehouse. In such a case, computer science can be directly connected to operations research. On the other hand, theorem proof research cannot replace empirical research for the large number of systems that are subject to human interference.

Experimental research can also support theoretical research because it requires exact model descriptions. For example, in the first paper on the design of Infogame (Casimir 1986), the characteristics of management games were used for a classification of information systems. An example in this thesis is the definition of a quality index in section 5.3.2. The notion that quality metrics for system development phases are different from quality metrics for a project as a whole is not obvious from the literature reviewed in section 2.2.

6.2.4 Engineering research

Engineering research creates new models and tools than can be tested in the laboratory or in the field. Accordingly, we may expect that engineering research stimulates empirical research. In information systems development, however, many engineering researchers are so convinced of the quality of their products that they see no need for empirical tests. Laboratory research can be applied in an earlier phase of the engineering research cycle than field research because it does not require a marketable product, but sometimes the quality of a new model or tool can only be assessed in the field, for example when a long learning period is needed before its benefits become apparent.

On the other hand, engineering research can support experimental research by the design of new experimental instruments. In this respect, the design of Infogame belongs to the domain of engineering research. However, experimental information systems research often has to use commercially available tools, and in some instances, such as CASE tools, even those may be too expensive for research purposes. In contrast, in such fields as physics or computer science, experimental research often uses specially designed instruments that are years ahead of commercially available apparatus. In some fields of the discipline of information systems, there may be a vicious circle: academic information systems departments are not awarded powerful tools because they do not produce results, and they do not produce results because they lack the tools.

6.2.5 Field research

Field research is the only type of research that can replace, rather than supplement, experimental research. It is appropriate when information systems development depends on contingency factors that cannot be adequately represented in the laboratory. Such factors are present when scaling down in time or in size delivers qualitatively different results. For example, the human relations that develop during years in the real-life working place cannot be emulated in a game that spans only a few days. In such cases, however, a field test may be also be unfeasible

because it is unethical or too expensive, and surveys may deliver inconclusive results because there are too few comparable cases.

Accordingly, though the need for field research may be invoked to criticize experimental research, few examples of actual field research in information systems development have been shown. Moreover, information systems development research offers opportunities for combining laboratory and field testing, because an information system that is used in a laboratory test, for example a program for investment advice in a stock exchange game, may be designed in a regular information systems department.

6.3 New uses of Infogame

6.3.1 Infogame changes and subdisciplines

Doke and Barrier (1994) showed there is no consensus on the taxonomy of the discipline of information systems. This is also shown by the difference between information systems management issues and research areas (Galliers 1993). To identify relevant subdisciplines, Fig. 6.1 shows the well known information systems pyramid first proposed by Head (1967).

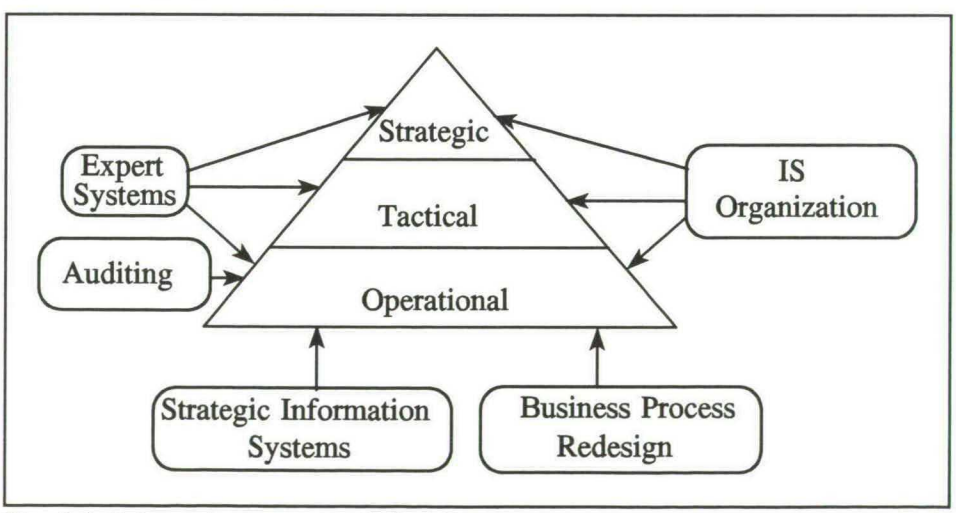


Fig. 6.1: Information systems subdisciplines

Traditional classifications (Doke and Barrier 1994) distinguish information systems on the *strategic, tactical, and operational level*. In addition, five newer subdisciplines are identified. *Expert systems* can be used to supplant decision makers at all levels. *Auditing*, including its subdiscipline *EDP auditing*, studies the reliability and efficiency of operational information systems. *Strategic information systems* is concerned with the strategic opportunities of information systems and *business process redesign* investigates the interaction between changes in organization and information system changes. The attention for the latter three subjects partially compensates for the traditional lack of interest in operational information systems from the research community. We use the term *IS organization* for the study of the communication between information systems at different levels, a subject that has some links to *group decision support systems* (GDSS) and *information systems planning*, but is not acknowledged as a field in its own right.

Presently, Infogame is used for research on the design and use of information systems for top management, and there is still considerable scope for further research in this field. Both for research in this area and for research in one of the five above-mentioned newer subdisciplines, Infogame can be changed on different levels, which are described in section 6.3.2.

The use of embedded expert systems, and the introduction of errors that make reliability an important issue in Infogame, are discussed from the point of view of the experimenter and educator in sections 6.3.3 and 6.3.4 respectively. Both were already included in the first proposals for Infogame (Casimir 1986). The topics *strategic information systems*, *business process redesign*, and *IS organization* are addressed together in section 6.3.5. This section shows that Infogame is unsuitable for some themes in information systems research.

The uses of Infogame and the desired changes depend on the *views* of the *sponsor*. As related in section 2.1, there are widely diverging views on the purpose and nature of information systems, which usually are based on conflicting views of the world as a whole. Infogame can be adapted to the needs of users with different views, but changes will only be made if the pertinent sponsor emerges. Accordingly, though in sections 6.3.2 through 6.3.5 many technical details are discussed, the choice between techniques should never be based on technical merits alone.

6.3.2 Levels of change

6.3.2.1 Introduction

Infogame can be changed on one of the four levels listed in Table 6.1. A game administrator who wants to make changes on the organization level only needs the general knowledge of Infogame provided by the Users Manual. To effect changes on the data level, the syntax and semantics of data should be studied from the Game Administrator's Manual. Changes of the interface program require knowledge of internal Infogame files, which are also described in the Game Administrator's Manual, and changes in the simulation program are not possible without the help of the designer.

Table 6.1: Changes in Infogame

Changes in	Examples
Organization	Number of teams, team composition, team tasks, time between rounds, use of standard packages.
Data	Market characteristics, production characteristics (e.g. multilevel product structure).
Interface program	Graphics output, multimedia, input screening.
Simulation program	Warehouses, trade in semi-finished products, errors in operational information system.

6.3.2.2 Changes in organization

Organization of a game entails defining the number, composition, and task of player teams, determining playing schedules, providing player instructions, and managing the distribution and processing of Infogame input and output.

Player teams

Until now, Infogame has been played by university students in information systems and business administration in the Netherlands. When it is offered to new groups, such as managers, information systems designers, vocational students, or students from other countries, results can be generalized and differences can be traced. New groups may, however, also require data that represent a more familiar environment or even a different interface. For example, students at Tilburg University played Infogame with an interface in the Dutch language that omitted such options as choosing more than one supplier for a material.

In most experiments, Infogame players both designed an information system and made business decisions. Assigning the two tasks to different teams of players may have decided advantages, especially if the relation between the teams portrays real-world relations. For example, the experiment described in section 5.2.6 could be repeated with information systems students acting as designers and real-world managers playing the role of Infogame managers.

Player teams may also be structured by making a distinction between top managers, who make strategic decisions, such as investments, once a year, and lower-level managers who make the more mundane production and marketing decisions every quarter.

Playing schedule

The time allowed for input in Infogame can be as short as five minutes or as long as one week, but real time playing is not possible in the present version. It is discussed in section 6.3.2.4. If Infogame is played with one hour or less between rounds, all information systems must be ready and tested before the game starts. If there is a day or more between successive rounds, players can update their information system between rounds, and write special purpose programs to handle unexpected problems. They can also defy the idea that Infogame output should be handled mechanically, and base decisions directly on the detailed output files.

Output for players

The game administrator determines the media used for input to and output from Infogame. Because in the present version all input and output files are in ASCII format, all types of E-Mail can be used as well as LAN or diskette. The choice of media depends on local conditions. For example, in one session at Tilburg university, diskettes were used because the security of the LAN was insufficient to prevent espionage and sabotage, and the Infogame input program could not be used on the server that permitted the use of E-Mail.

The game administrator can also determine which report files are delivered to players. For example, he can withhold the market and labour reports and he can give copies of competitors' files to players. He can also provide standard packages for data processing or provide the information delivered by such programs. For example, in one of our experiments, players received standard financial reports. If Infogame is used in an information systems development course, its use must be tuned with the systems development life cycle. For example, the game can be played once with the prototype systems and replayed later with the complete systems. It is also clear that the reliability of the information systems used by players is more important if the time between rounds is shorter.

6.3.2.3 Data changes

Infogame offers extensive opportunities for changing data. The game administrator can change the number and type of industries and the size of the consumer and labour markets. Because the names and characteristics of suppliers, machines and technologies can be changed, the level of abstraction can be reduced. A special feature is the possibility to incorporate complex product structures by defining materials that must be produced internally.

Data changes should be carefully planned, executed, and tested. Infogame detects all input errors from players, but is not immune to errors made by the game administrator.

6.3.2.4 Interface program changes

State-of-the-art adventure games show that interface changes such as the introduction of enhanced graphical output and multimedia make a game more attractive. This will elicit more realistic reactions and attract new groups, including players with less formal education. In the design of Infogame, however, designing a pretty interface had no priority, and moreover, in previous versions, the interface program shared data structures and internal files with the simulation program, and hence had to be programmed in Turbo-Pascal for MSDOS.

Because in the present version all communication between interface and simulation programs is effected by ASCII files, this restriction has been removed, and independent input programs can be constructed. For example, in a software engineering course, students can design an interface program that incorporates an expert system that checks input and suggests alternatives.

Real Time Gaming requires a major change in both the interface and the simulation program, because communication by files must be replaced by communication by messages. Its value, however, is arguable because in a real-time game, everyday decisions must be made in seconds and this may qualitatively alter the decision process.

6.3.2.5 Simulation program changes

For a classification of possible changes to the Infogame simulation program, we start from the model in Fig. 6.2. Presently, the simulation program contains two markets and several types of processes. All players take on the role of managers. The simulation model can be extended by changing existing processes and markets, by introducing new processes and markets, by adding players with other roles, and by establishing communication between players. In Fig. 6.2, new entities are represented by shaded blocks and new communication lines by dashed lines. Some examples of the extensions are given below. Apart from the specific advantages of each proposal, all changes diminish the possibility that players will just copy information systems from an earlier player generation. For most changes in the

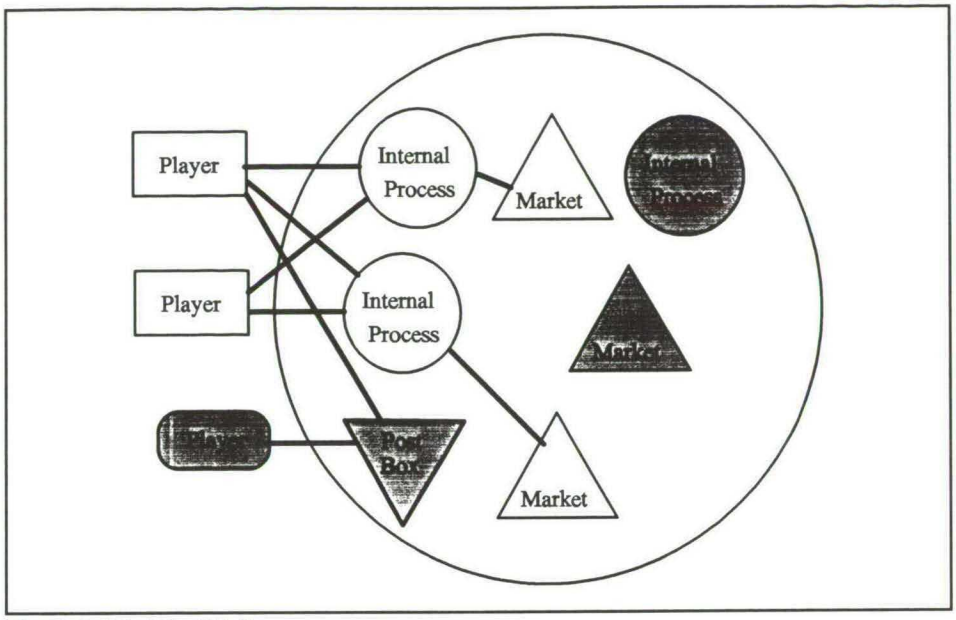


Fig 6.2: Model of Infogame

simulation program, the interface program and the data must be changed as well, either by the introduction of new instructions and data, or by a change in syntax or semantics of existing instructions and data.

Two possible extensions of the simulation program are intimately connected with specific applications. Insertion of *programmed rules* is needed for embedded expert systems and hence will be covered in section 6.3.3. *Uncertainty* about the results of internal processes primarily increases the utility of auditing and consequently it is discussed in section 6.3.4. This type of uncertainty should be distinguished from uncertainty about market results, which increases the importance of strategic and tactical information systems.

Changes in processes and markets

Few changes in processes and markets only affect a single process or a single market. An example is a change in the relation between number of workers, batch size and production time, which only affects the production process. On the

other hand, the introduction of sales workers that must be employed for every sale concerns the employment and sales processes and the labour and consumer markets.

New processes and markets

Examples of new processes are a maintenance process that employs repairmen for preventive maintenance and repairs and a storage process that operates a warehouse to store and retrieve materials and finished products. Such processes offer the opportunity to build an information system for a supporting process. When the number of processes becomes large, players will have to choose for which processes information systems will be built, which raises the issue of information systems planning.

Examples of new markets are markets for machines and materials. Presently, all machines and materials needed can be bought, whereas on a market, firms have to compete for resources. An increase in the number of markets increases uncertainty, which can be countered by buffers or by the use of information. In this context, the term *market* is only used for a market where one of the parties is simulated. A market where both suppliers and consumers are players, such as a stock market, is considered a communication system.

Introducing new markets at the supply side of Infogame makes no sense if one shares the view heralded by some extreme JIT proponents that all markets are strictly buyer markets, which precludes any uncertainty about deliveries. According to this view, the labour market should also be removed from Infogame, because the necessary number of workers can always can be engaged at current rates and workers can be arbitrarily dismissed at well.

Players with new roles

Presently, all players take on the role of managers. Apart from the distribution of tasks within teams, players with quite different roles can be introduced. The prime example is the role of the shareholder, who trades shares on the stock market and decides on takeovers. Shareholders need financial information,

and the relation between financial and management information is an interesting subject for further study.

Other roles are those of agencies that demand information on specific issues, such as the environment, labour relations or product quality. Such agencies will force players to design special purpose information systems that may or may not be integrated with management information systems. Because such agencies can be compared to auditors, their introduction also requires the type of uncertainty that is discussed in section 6.3.4.

Communication between players

Opportunities for transactions between players entail a change in Infogame organization as well as a change in programs and data, because players must have the time and opportunity for communication. For example, if trading of semi-finished products between companies is allowed, conditions such as prices, amounts and fines for nondelivery must be stored by the simulation program, but there must also be time for informal negotiations. Decisions that depend on communication with other parties often require building ad-hoc information systems, and this further lengthens the minimum time between rounds.

6.3.3 Expert systems

In Infogame, expert systems can be used at two levels. First, as discussed in section 6.3.2.4, they can be used in the interface program to advise on player decisions and to screen inputs. Second, they can be embedded into the Infogame program to simulate the behaviour of middle managers. For example, an expert system may decide which of several waiting production orders must be executed.

Such an expert system must be written in an existing or newly created programming language or expert system shell. The program or script should be translated by the interface program and executed or interpreted in the simulation program. During translation both syntactic and semantic errors should be detected because the simulation program could crash on encountering an erroneous expert.

Designing and implementing such a language is a major software engineering task and the use of embedded expert systems also exacts considerable effort from players. It is only worthwhile if control of internal processes is considered important for organizational success. A decision to extend Infogame with embedded expert systems corresponds with the view that middle management mainly executes programmed tasks, and that the performance of an organization can be significantly increased by improving the programs for those tasks. On the other hand, if solving minor unforeseen problems is considered the main task of middle management, embedded expert systems add little value, because solutions to such problems cannot be programmed.

6.3.4 Auditing

The audit process determines the correctness of accounting data by comparing data from different sources. For example, it determines whether the ending stock of a product is equal to beginning stock plus production minus sales, and if it is not, whether this is caused by theft, shrinkage, or incorrect recording of sales or production. When the cause of a problem is found, auditors will suggest a procedure to prevent its occurrence in the future, which must be implemented by management.

In the present version of Infogame, there is no need for an audit, because there is no difference between the plan for an event, the event itself, and the record of that event. For example, the amount on a suppliers invoice is always exactly equal to the amount ordered and the amount delivered. To implement the variations that call for an audit, we propose to introduce *daemons* into Infogame. A daemon is a random mechanism that changes a variable at unpredictable moments. For example, a cash daemon reduces the cash level, and a billing daemon causes errors in customer invoices. The notion of a daemon is universal. Daemon activity encompasses fraud and embezzlement, human errors, machine failures, epidemics, strikes and acts of God. If the activity of a daemon is detected, its recurrence can be prevented by appropriate management policy, but installing a preventive policy against all potential daemon activity is too costly.

Consequently, to prevent daemon activity, an accounting system must determine its level. Moreover, because daemons can corrupt accounting records, the accounting system itself should be audited by comparing data from different sources. For example, the stock level computed from input and output data should be compared to the stock level computed from inventory data. The present output of Infogame already contains some data items that can be used for auditing purposes.

Procedures to reduce daemon activity should be selected from a large number of options. Modelling those procedures is a major task in auditing research, and consequently, the transformation of Infogame to a tool for research and education in auditing needs a sponsor from the auditing discipline.

A subdiscipline of auditing is the field of *EDP auditing*, which examines the reliability and efficiency of automated information systems. EDP audits can be applied to the information systems built in the present version of Infogame, but they would be more appropriate for the accounting systems described in this section.

6.3.5 Information systems and organization

Installing strategic information systems entails a change in the business itself, and it may be driven by business needs (Earl 1992). The phrase "business process redesign" embodies this change, and hence we assume that strategic information systems are special instances of BPR. Research and education in BPR cannot be implemented in Infogame, because its business model cannot be changed by players. However, we still want to examine whether any type of game can be applied to BPR. Two types of game can be envisaged. First, gaming can be used to determine opportunities for and resistance against BPR in a specially prepared gaming model of an organization. Such a game would involve different human role players and hence could be labelled a social simulation. Second, a game could be planned where human roles are simulated, and players can change the tasks of employees. The design of such a game requires adequate human behaviour models, which demands a major research effort. A game that is easy to design would allow students to choice business processes from a number of options. However, such a game would be trivial if it guaranteed success to players opting for famous

examples, and its design would be difficult if it involved complex interactions between processes. In view of the difficulties of game design for BPR, we think that at the moment field research is a more appropriate strategy for BPR than laboratory research.

Infogame can, however, be used in research and education on IS organization. To this end, a choice of standard operational packages with extensive option lists should be provided for users. As a default, players would be provided with *ineffective* and *inefficient* information systems. This enables players to decide at which level a decision must be made. For example, in a multi-product company, a player should be able to choose to simulate either a production manager and a sales manager for all products, or a product manager for each product. The quality of decision making would then depend on the match between simulated tasks and the information available for each task. In this example, appointing product managers while providing information for sales and production managers should lead to disastrous results.

By selection of packages and options, players also implement an information systems policy because budgetary restraints will force them to accept at least some inferior default systems (Ribbers 1990). Costs and benefits of all types of information systems should be carefully tuned to prevent any bias towards some type of information system.

Such a version of Infogame is quite different from the present version described in chapter 4. Therefore, its implementation calls for a major design effort, and it cannot be implemented without a sponsor. An important design decision in the proposed version is which middle managers should be simulated by computer and which should be played by human beings. Computer simulation of important roles diminishes realism, whereas too many human players may induce a preponderance of role-playing over rational decision making.

6.4 Conclusions

It is one of the advantages of modern word-processing programs that it was easy to confirm that, until now, except for the preface and a quotation, the word "I" was never used in this thesis. In this last section, the word "I" is deliberately used to stress that it contains personal views, rather than scientific truth.

The principal conclusion from the previous sections of chapter 6 is that views on organizations and information determine research strategies, preferred information system topics, and desired attributes of Infogame. On the other hand, Infogame itself is a tool that can be used for widely divergent purposes.

Accordingly, Infogame research can proceed in two directions. First, it can be used to support research based on my own view of information systems, which entails that information is one way to reduce uncertainty. Second, it can be adapted to the needs of a sponsor with any view that is legally and morally acceptable.

Sections 6.3.3. and 6.3.4 showed that expert systems design and auditing pose the most interesting software engineering challenges. However, I do not think these are the most appropriate fields for further study from an information systems perspective, because I question the viability of organizations governed by expert systems, and I do not think auditing theoreticians are inclined to specify exact models of control procedures for use in a management game. More generally, I think that stressing the software engineering aspect of Infogame, or of other tools, such as CASE tools or compilers, may direct research into designing products that are not really needed, whereas the design of tools that are used in the real world is left to hobbyists and commercial firms.

Thus, though I started my research on Infogame in the role of the software engineer depicted in section 3.1, I took on the role of an experimenter researching the topics related in chapter 2 because no sponsor could be found. This led to research into information system design and use that was as rewarding as the design of Infogame. Meanwhile, the time spent by students in designing and using information systems for Infogame has surpassed the time spent to design and build Infogame itself, and consequently Infogame is no longer an experimental tool but a tool for experiments.

References

- Alavi,M. and P. Carlson, "A review of MIS research and disciplinary development", *Journal of MIS*, Vol 8 No 4 (Spring 1992), p 45-62.

- Andlinger,G.R., "Business games, play one", *Harvard Business Review*, Vol 36 No 2 (March/April 1958), p 115-125

- Anthonisse,J.M., Van Hee,K.M. and J.K. Lenstra, *Resource-constrained project scheduling: an international exercise in DSS development*, CWI, Amsterdam, 1987.

- Apt,K.R. and E.-R. Olderog, *Programmverifikation* (Program verification), Springer-Verlag, Berlin, 1994.

- Arnett,K.P and M.C. Jones, "Firms that choose outsourcing: a profile", *Information and Management*, Vol 26 (1994), p 179-199

- Avgerou,C. and T. Cornford, "A review of the methodologies movement", *Journal of Information Technology*, Vol 5 (1993), p 277-286.

- Avison,D.E. and G. Fitzgerald, *Information systems development: methodologies, techniques and tools*, Blackwell Scientific Publications, Oxford, 1988.

- Bailey,J.E. and S.W. Pearson, "Development of a tool for measuring and analyzing computer user satisfaction", *Management Science*, Vol 29 No 5 (May 1983), p 530-545.

- Bailin,S.C., "An object oriented requirements specification method", *Communications of the ACM*, Vol 32 No 5 (May 1989), p 608-623.

- Baker,F.T., "Chief programmer team management of production programming", *IBM Systems Journal*, 1972 No 1, p 56-73.

- Bedell,E.F., *The computer solution, strategies for success in the information age,* Dow Jones-Irwin, Homewood Ill, 1985.

- Bemelmans,T.M.A., *Bestuurlijke informatiesystemen en automatisering, 2nd ed.* (Management information systems and automation), Stenfert Kroese, Leiden, 1984.

- Ben-Ari,M., *Principles of concurrent programming,* Prentice Hall, Englewood Cliffs NJ, 1982.

- Benbasat,I., "An analysis of research methodologies", in F.W. McFarlan (ed), *The information systems research challenge*, Harvard Business School Press, Boston, 1985.

- Benbasat,I. and R.G. Schroeder, "An Experimental Investigation into some MIS Design Variables", *MIS Quarterly,* Vol 1 No 1 (March 1977), p 37-50.

- Benbasat,I. and A.S. Dexter, "An experimental evaluation of graphical and color-enhanced information presentation", *Management Science*, Vol 31 No 11 (November 1985), p 1348-1364.

- Benbasat,I. and A.S. Dexter, "An investigation of the effectiveness of color and graphical information presentation under varying time constraints", *MIS Quarterly*, Vol 10 No 1 (March 1986), p 58-83.

- Benbasat,I., Dexter,A.S. and P. Todd, (1986a) "The influence of color and graphical information presentation in a managerial decision simulation", *Human-Computer Interaction*, Vol 2 (1986), p 65-92.

- Benbasat,I., Dexter,A.S. and P. Todd, (1986b) "An experimental program investigating color-enhanced and graphical information presentation: an integration of the findings", *Communications of the ACM*, Vol 29 No 11 (November 1986), p 1094-1105.

- Benbasat,I. and B.R. Nault, "An evaluation of empirical research in managerial support systems", *Decision Support Systems*, Vol 6 (1990), p 203-226.

- Benjamin,R.I., *Control of the information system development cycle*, Wiley-Interscience, New York, 1971.

- Berne,E., *Games people play,* Grove Press, New York, 1967.

- Bingham,R.S.jr, *Quality control handbook, 2nd ed.* (edited by J.M. Juran, L.A. Seder and F.M. Gryna), Mc Graw-Hill, New York, 1962.

- Blokdijk,A. and P. Blokdijk, *Planning and design of information systems,* Academic Press, London, 1989.

- Boar,B.H., *Application prototyping*, John Wiley & sons, New York, 1984.

- Boehm,B.W., "Seven basic principles of software engineering", *The Journal of Systems and Software*, Vol 3 (1983), p 3-24.

- Boehm,B.W., *Software Engineering Economics,* Prentice Hall, Englewood Cliffs NJ, 1981.

- Booch,G., "Object-oriented development", *IEEE Transactions on Software Engineering*, Vol SE-12 No 2 (February 1986), p 211-221.

- Botermans,J., Visser,N. and N. Dekker, *Timpen, hinkelen en pierebollen*, Unieboek, Houten, 1991.

- Brandes,D. and H. Phillips, *Gamesters' Handbook, 140 games for teachers and group leaders*, Hutchinson, London, 1979.

- Brinch Hansen,P., *Operating system principles,* Prentice Hall, Englewood Cliffs NJ, 1973.

- Brinkkemper,S., *Formalisation of information systems modelling,* Ph D. Thesis, Nijmegen, 1990.

- Brooks,F.P, "No silver bullet, essence and accidents of software engineering", *Computer*, Vol 20 No 4 (April 1987), p 10-19.

- Bubenko,J., Langefors,B. and A. Sølvberg (eds), *Computer-aided information systems analysis and design*, Studentlitteratur, Lund, 1971.

- Buffa,E.S., *Modern production/operations management, 7th ed.,* John Wiley & sons, New York, 1983.

- Burch,J.G., Strater,F.R. and G. Grudnitski, *Information systems: theory and practice, 2nd ed.*, John Wiley & sons, New York, 1979.

- Burgess,T., "Learning lessons from business games: factors influencing software development", *British Journal of Educational Technology*, Vol 25 No 2 (1994), p 113-124.

- Caillois,R., *Man, play and games*, Schocken books, New York, 1979. (translation of: *Les jeux et les hommes*, Gallimard, Paris, 1958)

- Canning,R.G., "The advent of structured programming", *EDP Analyzer*, Vol 12 No 6 (June 1974), p 1-14.

- Casimir,R.J., "DSS, information systems, and management games", *Information and Management*, Vol 11 No 3 (October 1986), p 123-129.

- Chatfield,M., *A history of accounting thought*, Robert E. Krieger Publishing Company, Huntington, N.Y., 1977.

- Chen,P.P., "The entity-relationship model - toward a unified view of data", *ACM Transactions on Database Systems*, Vol 1 No 1 (March 1976), p 9-36.

- Cheon,M.J., Grover,V. and R. Sabherwal, "The evolution of empirical research in IS, a study in IS maturity", *Information and Management*, Vol 24 (1993), p 107-119.

- Christensen,L.B., *Experimental methodology, 4th ed.,* Allyn and Bacon Inc, Boston, 1988.

- Cohen,B., "Justification of formal methods for system specification" and "Rejustification of formal methods for system specification", *Software Engineering Journal*, Vol 4 No 1 (Januari 1989), p 26-38.

- Cohen,K.J., Cyert,R.M., Kuehn,A.E., Miller,M.H., Van Wormer,T.A., and P.R. Winters, "The Carnegie tech management game", in: Guetzkow, H. (ed), *Simulation in social science*, Prentice Hall, Englewood Cliffs NJ, 1962.

- Cohen,K.J. and E. Rhenman, "The role of management games in education and research", *Management Science*, Vol 7 (1960/61), p 131-166.

- *Collins Cobuild english language dictionary*, Collins, London, 1987.

- Cooper,R.B., "Review of management information systems research: a management support emphasis", *Information Processing and Management*, Vol 24 No 1 (1988), p 73-102.

- Coppieters,D., *An intelligent gaming environment*, Delft University Press, Delft, 1990.

- Courtney,J.F., Bierer,J.M., Luckew,T.G. and J.J. Kabbes, "Using management games as an aid to teaching MIS design", *Decision Sciences*, Vol 9 (1978), p 496-509.

- Courtney,J.F., DeSanctis,G. and Kasper,G.M., "Continuity in MIS/DSS Laboratory Research: The Case for a Common Gaming Simulator", *Decision Sciences*, Vol 14 No 3 (July 1983), p 419-439.

- Cox,B.J., "There is a silver bullet", *Byte*, Vol 15 No 10 (October 1990), p 209-218.

- Daellenbach,H.G., "Alternative OR", *OR/MS Today*, Vol 21 No 6 (December 1994), p 44-49.

- Daniels,A. and D. Yeates (eds), *Basic training in systems analysis*, Sir Isaac Pitman & Sons, London, 1969.

- Davenport,T.H. and D.B. Stoddard, "Reengineering : business change of mythic proportions?", *MIS Quarterly*, Vol 18 No 2 (June 1994), p 121-127.

- Davis,G.B., *Management information systems, conceptual foundations, structure and development,* McGraw-Hill, New York, 1974.

- Dean,B.V. and M.J. Nishry, "Scoring and profitability models for evaluating and selecting engineering projects", *Operations Research*, Vol 13 (1965), p 550-569.

- DeFanti,T.A., "The mass impact of videogame technology", in: *Advances in Computers*, Vol 24 (edited by M.C. Yovits), Academic Press, Orlando, 1984.

- De Jager,J.L., *Volksgebruiken in Nederland* (Folk customs in the Netherlands), Het Spectrum, Utrecht, 1981.

- Delen,G.P.A.J. and D.B.B. Rijsenbrij, (1990a) "Kwaliteitsattributen van automatiseringsprojecten en informatiesystemen" (Quality attributes of automation projects and information systems), *Informatie*, Vol 32 No 1 (January 1990), p 46-55.

- Delen,G.P.A.J. and D.B.B. Rijsenbrij, (1990b) "Het 'realiseren' en 'meten' van produktiekwaliteit" (Realizing and measuring production quality), *Informatie*, Vol 32 No 11 (November 1990), p 858-867.

- DeLone,W.H. and E.R. McLean, "Information systems success: the quest for the dependent variable", *Information Systems Research*, Vol 3 No 1 (1992), p 60-95.

- DeMarco,T., *Controlling software projects*, Yourdon Press, New York, 1982.

- Dennis,A.R., Nunamaker,J.F. and D.R. Vogel, "A comparison of laboratory and field research in the study of electronic meeting systems", *Journal of MIS*, Vol 7 No 3 (Winter 1990-91), p 107-135.

- DeSanctis,G., "Computer graphics as decision aids: directions for research", *Decision Sciences*, Vol 15 (1984), p 463-487.

- De Troyer,O.M.F., *On data schemata transformation*, Ph.D. Thesis, Tilburg University, 1993.

- De Wit,D., *The shaping of automation, a historical analysis of the interaction between technology and organization, 1950-1985*, Verloren, Hilversum, 1994.

- Dickson,G.W., Senn,J.A. and N.L. Chervany, "Research in management information systems: the Minnesota experiments", *Management Science*, Vol 23 No 9 (May 1977), p 913-923.

- Dijkstra,E.W., "Cooperating sequential processes", in: Genuys,F. (ed), *Programming languages*, Academic Press, New York, 1968.

- Dijkstra,E.W., "The structure of the THE multiprogramming system", *Communications of the ACM*, Vol 11 No 5 (May 1968), p 341-346.

- Dijkstra,E.W., "The humble programmer", *Communications of the ACM*, Vol 15 No 10 (October 1972), p 859-866.

- Dillon,A., "Reading from paper versus screens: a critical review of the empirical literature", *Ergonomics,* Vol 35 No 10 (1992), p 1297-1326.

- Doke,E.R.. and T. Barrier, "An assessment of information systems taxonomies: time to re-evaluate?", *Journal of Information Technology*, Vol 9 (1994), p 149-157.

- Downs,E., Clare,P. and I. Coe, *Structured systems analysis and design method, application and context*, Prentice Hall, New York, 1988.

- Dreyer,A., "Scoring-Modelle bei Mehrfachzielsetzungen, eine Analyse des Entwicklungsstandes von Scoring-Modelle", *Zeitschrift für Betriebswirtschaft*, Vol 44 (1974), p 255-274.

- Duke,R.D., "Management under complexity, gaming/simulation as a predecisional tool", *Simulation and Games*, Vol 13 No 3 (September 1982), p 365-373.

- Earl,M.J., "Putting IT in its place: a polemic for the nineties", *Journal of Information Technology*, Vol 7 (1992), p 100-108.

- Edwards,C. and J.W. Peppard, "Business process redesign : hype, hope or hypocrisy?", *Journal of Information Technology*, Vol 9 No 4 (December 1994), p 251-266.

- Eilon,S., "Measuring quality of information systems", *Omega*, Vol 21 No 2 (1993), p 135-138.

- Elfring,T. and G. Baven, "Outsourcing technical services: stages of development", *Long Range Planning*, Vol 27 No 5 (1994), p 42-51.

- Elgood,C., *Handbook of management games, 2nd ed.,* Gower, Aldershot, 1981.

- Elias,N., *Über den prozess der zivilisation, 2nd ed.*, Francke Verlag, Bern, 1969.

- Erasmus, *Moriae encomium, id est stultitiae laus* (Praise of folly), Basel, 1515, edition with Latin text and Dutch translation, Paris, Amsterdam, 1949.

- Estes,J.E., "But what will the workers do ?", *Simulation and Games*, Vol 17 No 2 (July 1986), p 245-262.

- Falkenberg,E.D. and P. Lindgreen (eds), *Information system concepts: an in-depth analysis*, North-Holland, Amsterdam, 1989.

- Friedman,A.L., *Computer systems development, history, organization and implementation,* John Wiley & Sons, Chichester, 1989.

- Galletta,D.F. and A.L. Lederer, "Some cautions on the measurement of user information satisfaction", *Decision Sciences*, Vol 20 (1989), p 419-438.

- Galliers,R.D., "In search of a paradigm for information systems research", in: E. Mumford et al. (eds), *Research methods in information systems,* North-Holland, Amsterdam, 1985.

- Galliers,R.D., "Research issues in information systems", *Journal of Information Technology*, Vol 8 (1993), p 92-98.

- Garceau,S., Oral,M. and R.J. Rahn, "The influence of data-presentation mode on strategic decision-making performance", *Computers and Operations Research*, Vol 15 No 5 (1988), p 479-488.

- Gatian,A.W., "Is user satisfaction a valid measure of system effectiveness?", *Information and Management*, Vol 26 (1994), p 118-131.

- Geurts,J.L.A., *Model en spel* (Model and game), Ph.D. thesis, University of Nijmegen, 1981.

- Geurts,J.L.A., *Omkijken naar de toekomst* (Looking back to the future), Samsom H.D. Tjeenk Willink, Alphen aan den Rijn, 1993.

- Gilb,T., *Software metrics*, Studentlitteratur, Lund, 1976.

- Goguen,J.A., "The dry and the wet", in: Falkenberg,E.D., Rolland,C. and E.N. El-Sayed (eds), *Information systems concepts: improving the understanding*, North-Holland, Amsterdam, 1992.

- Greenbaum,J., *In the name of efficiency*, Temple University, Philadelphia, 1979.

- Greenblat,C.S., "Communicating about simulation design", "Social issues, game design, and research: an introduction" and "CAPFJEFOS: the village development game" in: Crookall,D. et al. (eds), *Simulation-gaming in the late 1980s*, Pergamon Press, Oxford, 1987, p 23-33, 135-137, 159-163.

- Gremmen,H.J.F.M., *SIER, a macro-economic game on cooperation and conflict in international economics*, Ph.D. Thesis, Tilburg University, 1989.

- Gries,D., "Teaching calculation and discrimination: a more effective curriculum", *Communications of the ACM*, Vol 34 No 3 (March 1991), p 44-55.

- Griswold,R.E. and M.E. Griswold, *The ICON programming language*, Prentice Hall, Englewood Cliffs NJ, 1983.

- Gross,D. and C.M. Harris, *Fundamentals of queuing theory*, John Wiley & sons, New York, 1985.

- Grover,V., Cheon,M.J. and J.T.C. Teng, "A descriptive study on the outsourcing of information systems functions", *Information and Management*, Vol. 17 (1994), p 33-44.

- Grunfeld,F.V., Vié,L., Williams,G. and R.C. Bell, *Spelletjes uit de hele wereld* (Games from all over the world), Kosmos, Amsterdam, 1975.

- Guimaraes,T., "Managing application program maintenance expenditures", *Communications of the ACM*, Vol 26 No 10 (October 1983), p 739-746.

- Guimaraes,T., "A study of application program development techniques", *Communications of the ACM*, Vol 28 No 5 (May 1985), p 494-499.

- Guimaraes,T., Igbaria,M. and M. Lu, "The determinants of DSS success: an integrated model", *Decision Sciences*, Vol 23 (1992), p 409-430.

- Hall,A., "Seven myths of formal methods", *IEEE Software*, Vol 7 No 5 (September 1990), p 11-19.

- Hamilton,S. and B. Ives, "MIS research strategies", *Information and Management*, Vol 5 (1982), p 339-347.

- Hammer,M., "Reengineering work : don't automate, obliterate", *Harvard Business Review*, Vol 68 No 4 (July/August 1990), p 104-112.

- Harel,E.C. and E.R. McLean, "The effects of using a nonprocedural computer language on programmer productivity", *MIS Quarterly*, Vol 8 No 2 (June 1985), p 109-120.

- Hares,J.S., *SSADM version 4, the advanced practitioner's guide*, John Wiley & sons, Chichester UK, 1994.

- Hartman,W., Matthes,H. and A. Proeme, *Information systems handbook: Analysis, Requirements definition, Design and development, Implementation and evaluation*, Philips Data Systems, Apeldoorn, 1968.

- Head,R.V., "Management information systems: a critical appraisal", *Datamation*, May 1967, p 22-27.

- Hertz,D.B. and P.G. Carlson, "Selection, evaluation and control in R & D projects", in: B.V. Dean (ed), *Operations research in research and development*, John Wiley & Sons, New York, 1963.

- Hice,G.F., Turner,W.S. and L.F. Caswell, *System development methodology*, North-Holland, Amsterdam, 1974.

- Hirschheim,R. and H.K. Klein, "Four paradigms of information systems development", *Communications of the ACM*, Vol 32 No 10 (October 1989), p 1199-1216.

- Hoare,C.A.R., "An axiomatic basis for computer programming", *Communications of the ACM*, Vol 12 No 10 (October 1969), p 576-580.

- Holsapple,C.W., Johnson,L.E., Manakyan,H. and J. Tanner, "Business computing system research: structuring the field", *Omega*, Vol 22 No 1 (1994), p 69-81.

- Holt,C.C., "Improving the labor market trade-off between inflation and unemployment", *American Economic Review*, Vol 59 No 2 (1969), p 134-146.

- Hopstaken,B. and A. Kranendonk, *Informatieplanning in tweevoud* (Information planning in duplicate), Stenfert Kroese, Leiden, 1990.

- Huber,G.P., "Organization science contribution to the design of decision support systems", in: Fick, G. and R.H. Sprague (eds), *Decision support systems: issues and challenges*, Pergamon Press, Oxford, 1980.

- Hughes,C.T. and M.L. Gibson, "Students as surrogates for managers in a decision-making environment: an experimental study", *Journal of MIS*, Vol 8 No 2 (Fall 1991), p 153-166.

- Huizinga,J., *Homo ludens*, Tjeenk Willink, Haarlem, 1938.

- Hwang,M.I.H. and B.J.P. Wu, "The effectiveness of computer graphics for decision support: a meta-analytical integration of research findings", *Data Base*, Vol 21 No 2/3 (Fall 1990), p 11-20.

- Iivari,J., "Implementability of in-house developed vs. application package based information systems", *Data Base*, Vol 21 No 1 (Spring 1990), p 1-10.

- Ives,B., "Graphical user interfaces for business information systems", *MIS Quarterly*, Vol 6 Special issue (December 1982), p 14-47.

- Jacobson,I., "Object oriented development in an industrial environment", *OOPSLA 87 Conference Proceedings, SIGPLAN Notices*, Vol 22 No 12 (December 1987), p 183-191.

- Janssen,H., *Geschiedenis van de speelkaart* (History of the playing card), Elmar, Rijswijk, 1985.

- Jarvenpaa,S.L., Dickson,G.W. and G. DeSanctis, "Methodological issues in experimental IS research: experiences and recommendations", *MIS Quarterly*, Vol 9 No 2 (June 1985), p 141-156.

- Jenkins,A.M., Naumann,J.D. and J.C. Wetherbe, "Empirical investigation of systems development practices and results", *Information and Management*, Vol 7 (1984), p 73-82.

- Jones,D.W., "An empirical comparison of priority-queue and event-set implementations", *Communications of the ACM*, Vol 29 No 4 (April 1986), p 300-311.

- Jordan,R., Smilan,R. and A. Wilkinson, "Streamlining the project cycle with object-oriented requirements", *OOPSLA 94 Conference Proceedings, SIGPLAN Notices*, Vol 29 No 10 (October 1994), p 287-300.

- Juran,J.M., *Juran on planning for quality*, The Free Press, New York, 1988.

- Kaplan,R.S. and D.P. Norton, "The balanced scorecard, measures that drive performance", *Harvard Business Review*, Vol 70 No 1 (January/February 1992), p 71-79.

- Kettinger,W.J., Grover,V., Guha,S. and A.H. Segars, "Strategic information systems revisited : a study in sustainability and performance", *MIS Quarterly*, Vol 18 No 1 (March 1994), p 31-58.

- King,W.R. and R. Sabherwal, "The factors affecting strategic information systems applications : an empirical assessment", *Information and Management*, Vol 23 No 4 (October 1992), p 217-235.

- Klabbers,J.H.G., "A user-oriented taxonomy of games and simulations", in: Crookall.D. et al. (eds), *Simulation-gaming in the late 1980s*. Pergamon Press, Oxford, 1987.

- Kleijnen,J.P.C., *Computers and Profits,* Addison-Wesley, Reading Mass., 1980.

- Kleijnen,J.P.C., "De rol van wiskundige modellen en technieken in de bestuurlijke informatica" (The role of mathematical models and techniques in information systems), *Informatie*, Vol 23 No 6 (June 1981), p 387-393.

- Kleijnen,J.P.C., "Analyzing simulation experiments with common random numbers", *Management Science*, Vol 34 No 1 (January 1988), p 65-74.

- Kleijnen,J.P.C., "Verification and validation of simulation models", *European Journal of Operational Research*, Vol 82 (1995), p 145-162.

- Kleijnen,J.P.C. and W. Van Groenendaal, *Simulation, a statistical perspective*, John Wiley & Sons, Chichester, 1992.

- Klein,H.K. and K. Lyyttinen, "The poverty of scientism in information systems", in: E. Mumford et al. (eds), *Research methods in information systems*, North-Holland, Amsterdam, 1985.

- Kotler,P., *Principles of marketing, 3d ed.*, Prentice Hall, Englewood Cliffs NJ, 1986.

- Kraft,P., *Programmers and managers, the routinization of computer programming in the united states*, Springer-Verlag, New York, 1977.

- Kruijswijk,K.W., *Bibliotheca Van de Linde-Niemeijeriana aucta et de novo descripta*, Staatsdrukkerij, Den Haag, 1974.

- Kuhn,T.S., *The structure of scientific revolutions, 2nd ed.,* The University of Chicago Press, Chicago, 1970.

- Laagland,P.M.T., *Modeling in information systems development*, Ph.D. Thesis, Free University Amsterdam, 1983.

- Laden,H.N. and T.R. Gildersleeve, *System design for computer applications*, John Wiley & sons, New York, 1963.

- Lapin,L., *Quantitative methods for business decisions, 4th ed.*, Harcourt Brace Jovanovich, San Diego, 1988.

- Launi,J.D., "A structured methodology for off-the-shelf software implementation", *Journal of Systems Management*, Vol 42 No 10 (October 1991), p 6-9.

- Lucas,H.C., "An experimental investigation of the use of computer-based graphics in decision making", *Management Science*, Vol 27 No 7 (July 1981), p 757-768.

- Lundeberg,M., Goldkuell,G. and A. Nilsson, *Information system development, a systematic approach,* Prentice Hall, Englewood Cliffs NJ, 1981.

- Marsden,J.R. and D.E. Pingry, "End user - IS design professional interaction - information exchange for firm profit or end user satisfaction?", *Information and Management*, Vol 14 (1988), p 75-80.

- McCracken,J.D., Weiss,H. and T.H. Lee, *Programming business computers*, John Wiley & Sons, New York, 1959.

- McFarlan,F.W., "Information technology changes the way you compete", *Harvard Business Review*, Vol 62 No 3 (May/June 1984), p 98-103.

- Meckel,J., *Studien über das Kriegsspiel* (Studies on war games), Ernst Siegfried Mittler und Sohn, Berlin, 1873.

- Melone,N.P., "A theoretical assessment of the user-satisfaction construct in information systems research", *Management Science*, Vol 36 No 1 (January 1990), p 76-80.

- Montazemi,A.A. and S. Wang, "The effect of modes of information presentation on decision making: a review and meta-analysis", *Journal of MIS*, Vol 5 No 3 (Winter 1988-89), p 101-127.

- Mottley,C.M. and R.D. Newton, "The selection of projects for industrial research", *Operations Research*, Vol 7 (1959), p 740-751.

- Mynatt,B.T., *Software engineering with student project guidance*, Prentice Hall, Englewood Cliffs NJ, 1990.

- Necco,C.R., Gordon,C.L. and N.W. Tsai, "Systems analysis and design: current practices", *MIS Quarterly*, Vol 11 No 4 (December 1987), p 461-475.

- Nolan,R.L., "Managing the computer resource: a stage hypothesis", *Communications of the ACM*, Vol 16 No 7 (July 1973), p 399-405.

- Nolan,R.L. and H.H. Seward, "Measuring user satisfaction to evaluate information systems", in: Nolan, R.L. (ed), *Managing the data resource function*, West Publishing Company, St Paul, 1974.

- Norris,D.R., "External validity of business games", *Simulation and Games*, Vol 17 No 4 (December 1986), p 447-459.

- Nunamaker,J.F., Chen,M. and T.D.M. Purdin, "Systems development in information systems research", *Journal of MIS*, Vol 7 No 3 (Winter 1990-91), p 89-106.

- Olle,T.W., Sol,H.G. and A.A. Verrijn Stuart (eds), *Information systems design methodologies: a comparative review*, North-Holland, Amsterdam, 1982.

- Olle,T.W., Hagelstein,J., Macdonald,I.A., Rolland,C., Sol,H.G., Van Assche,F.J.M. and A.A. Verrijn Stuart, *Information systems methodologies, a framework for understanding*, Addison-Wesley, Wokingham, England, 1988.

- Opie,I. and P. Opie, "Children games in street and playground", in: Bruner,J.S, Jolly,A. and K. Silva, eds, *Play, its role in development and evolution*, Basic Books, New York, 1976.

- Palvia,P. and J.T. Nosek, "A field examination of system life cycle techniques and methodologies", *Information and Management*, Vol 25 (1993), p 73-84.

- Parkinson,C.N., *Parkinson's Law or the Pursuit of Progress*, John Murray, London, 1958.

- Pfleeger,S.L., "Experimental design and analysis in software engineering, part 1 and 2" *Software Engineering Notes*, Vol 19 No 4 (October 1994), p 16-20, Vol 20 No 1 (January 1995), p 22-26.

- Rands,T., "The key role of applications software make-or-buy decisions", *Journal of Strategic Information Systems*, Vol 1 No 4 (September 1992), p 215-223.

- Rao,V.S. and S.L. Jarvenpaa, "Computer support of groups: theory-based models for GDSS research", *Management Science*, Vol 37 No 10 (October 1991), p 1347-1362.

- Remus,W., "Graduate students as surrogates for managers in experiments on business decision making", *Journal of Business Research*, Vol 14 (1986), p 19-25.

- Ribbers,P.M.A., "Informatiebeleid: het plannen van de invloed van informatietechnologie op de onderneming" (Information policy: planning the influence of information technology on the firm), *Bedrijfskunde*, Vol 62 No 2 (1990), p 141-151.

- Ricciardi,F.M., Craft,C.J., Malcolm,D.G., Bellman,R., Clark,C., Kibbee,J.M. and R.H. Rawdon, *Top management decision simulation*, American Management Association, New York, 1957.

- Ritchie,D.M., "Reflections on software research", *Communications of the ACM*, Vol 27 No 8 (August 1984), p 758-760.

- Ritchie,D.M. and K. Thompson, "The unix time sharing system", *Communications of the ACM*, Vol 17 No 7 (July 1974), p 365-375.

- Robey,D., Smith,L.A. and L.R. Visijayasarathy, "Perception of conflict and success in information systems development projects", *Journal of MIS*, Vol 10 No 1 (Summer 1993), p 123-139.

- Royce,W.W., "Managing the development of large software systems", in: *Proceedings of IEEE Wescon 1970*, reprinted in Thayer,R.H. (ed), *Software engineering project management*, Computer Science Press of the IEEE, Washington, 1988.

- Rubin,M.L., *Handbook of data processing management*, Vol 1: *Introduction to the system lifecycle*, Vol 2: *System life cycle standards*, Vol 3: Zuckerman, P., *System life cycle standards - form methods*, Brandon/Systems Press, Princeton, 1970.

- Saarinen,T., "System development methodology and project success", *Information and Management*, Vol 19 (1990), p 183-193.

- Sammet,J.E., *Programming languages: history and fundamentals*, Prentice Hall, Englewood Cliffs NJ, 1969.

- Scanlan,D.A., "Structured flowcharts outperform pseudocode: an experimental comparison", *IEEE Software*, Vol 6 No 5 (September 1989), p 28-36.

- Schneider,H.-J. and A.I. Wasserman (eds), *Automated tools for information systems design*, North-Holland, Amsterdam, 1982.

- Schroeder,R.G., *Operations management,* McGraw-Hill, New York, 1981.

- Scott Morton,M.S., *Management Decision Systems, Computer-Based Support for Decision Making*, Harvard University, Boston, 1971.

- Scott Morton,M.S., "The state of the art of research", in F.W. McFarlan (ed), *The information systems research challenge,* Harvard Business School Press, Boston, 1985.

- Sharda,R., Barr,S.H. and J.C. McDonnel, "Decision support system effectiveness: a review and empirical test", *Management Science*, Vol 34 No 2 (February 1988), p 139-159.

- Shneiderman,B., Mayer,R., McKay,D. and P. Heller, "Experimental investigations of the utility of detailed flowcharts in programming", *Communications of the ACM*, Vol 20 No 6 (June 1977), p 373-381.

- Shogan,A.W., *Management science*, Prentice Hall, Englewood Cliffs NJ, 1988.

- Shubik,M., *Games for society, business and war, towards a theory of gaming*, Elsevier, New York, 1975.

- Siskens,W.J.A., Heemstra,F.J. and H. Van de Stelt, "Kostenbeheersing bij automatiseringsprojecten, een empirisch onderzoek" (Cost control in automation projects, an empirical study), *Informatie*, Vol 31 No 1 (January 1989), p 34-43.

- Sommerville,I., *Software engineering, 4th ed.,* Addison-Wesley, Wokingham, England, 1992.

- Spivey,J.M., *The Z notation : a reference manual*, Prentice Hall, New York, 1989.

- Sprague,R.H. and E.D. Carlson, *Building Effective Decision Support Systems,* Prentice Hall, Englewood Cliffs NJ, 1982.

- Stevens,W.G.R., *Modular programming and management*, Pall Mall Press, London, 1969.

- Sumner,M., "An assessment of alternative application development approaches", *Information and Management*, Vol 10 (1986), p 197-206.

- Swinkels,G.J.P. and P.P.M.G.G. Brouwers, "Qualify: beoordeling effectiviteit en efficiëntie van informatiesystemen" (Qualify: assessment of effectiveness and efficiency of information systems), *Compact*, Vol 17 No 3 (Fall 1990), p 20-33.

- Tanenbaum,A.S., "Correctness proofs considered harmful or in defence of program testing", *SIGPLAN Notices*, Vol 11 No 5 (May 1976), p 64-68.

- Tardy,J.E., "Strategies for software acquisition", *Journal of Systems and Software*, Vol 18 No 3 (July 1992), p 281-285.

- Teach,R.D., "Profits: the false prophet in business gaming", *Simulation and Gaming*, Vol 21 No 1 (March 1990), p 12-26.

- Teichrow,D., *Automatisering systeemontwerp* (Automatization of system design, text in English), Stichting het Nederlands Studiecentrum, Amsterdam, 1970.

- Teng,J.T.C. and D.F. Galletta, "MIS research directions: a survey of researchers views", *Data Base*, Vol 21 No 2/3 (Fall 1990), p 1-10.

- Teory,T.J., Yang,D. and J.P. Fry, "A logical design methodology for relational databases using the extended entity-relationship model", *ACM Computing Surveys*, Vol 18 No 2 (June 1986), p 197-222.

- Ter Gouw,J., *De volksvermaken* (Folk amusements), Haarlem, 1871, Reprint 19XX.

- Thiers,J.B., *Traite de jeux et des divertissements, qui peuvent être permis, ou qui doivent être défendu au chrétiens selon les règles de l'église et le sentiment des pères* (Tract on the games and pastimes that can be allowed to or that should be forbidden to christians according to the church and the ideas of the church fathers), Dezallier, Paris, 1686.

- Thomas,C.T., "The genesis and practice of operational gaming", *Proceedings of the first international conference on operational research*, Operations Research Society of America, Baltimore, 1957, p 65-81.

- Thompson,K., "Reflections on trusting trust", *Communications of the ACM*, Vol 27 No 8 (August 1984), p 758-760.

- Turban,E., *Decision support and expert systems, 3d ed.*, Macmillan Publishing Company, New York, 1993.

- Underwood,E.A., "Nightingale, Florence", in *Encyclopaedia Brittanica, Macropaedia, 15th ed.*, Vol 13, p 99-100, Encyclopaedia Brittanica, Chicago, 1974.

- Van den Herik,H.J., *Computerschaak, schaakwereld en kunstmatige intelligentie* (Computer chess, chess world and artificial intelligence), Academic service, 's Gravenhage, 1983.

- Van der Genugten,B., *Blackjack in Holland casino's: hoe de dealer te verslaan* (Blackjack in Holland casinos: how to beat the dealer), Tilburg University Press, Tilburg, 1993.

- Van der Genugten,B. and P.E.M. Borm, "Golden ten: een kans- of een behendigheidsspel" (Golden ten, game of chance or game of strategy), *Kwantitatieve Methoden*, Vol 38 (1991), p 61-80.

- Van der Meer,F.B. and T. Roodink, "The dynamics of automation: a structural constructionist approach", *Informatization and the Public Sector*, Vol 1 (1991), p 121-141.

- Van der Pijl,G.J., *Kwaliteit van informatiesystemen in theorie en praktijk* (Quality of information systems in theory and practice), Ph.D. Thesis, Tilburg University, 1993.

- Van Horn,R.L., "Empirical studies of management information systems", *Data Base*, Vol 5 (1973), p 172-180.

- Van Schaik,F.D.J., *Effectiveness of decision support systems,* Delft University Press, Delft, 1988.

- Van Steenis,H., *How to plan, develop and use information systems,* Dorset House Publishing, New York, 1990.

- Vennix,J.A.M., *Mental models and computer models*, Ph.D. Thesis, University of Nijmegen, 1990.

- Verhulst,A., *Spellen gespeeld en gewogen* (Games played and weighed), Spectrum, Utrecht, 1985.

- Verrijn Stuart,A.A., "Some reflections on the namur conference on information systems concepts", in: Falkenberg, E.D. and P. Lindgreen (eds), *Information system concepts: an in-depth analysis*, North-Holland, Amsterdam, 1989.

- Vessey,I., "Cognitive fit: a theory-based analysis of the graph versus tables literature", *Decision Sciences*, Vol 22 (1991), p 219-241.

- Vogel,D.R. and J.C. Wetherbe, "MIS research: a profile of leading journals and universities", *Data Base*, Vol 16 No 1 (Fall 1984), p 3-14.

- Vogel,D.R. and J.F. Nunamaker, "Group decision support system impact: multi-methodological exploration", *Information and Management*, Vol 18 (1990), p 15-28.

- Vogel,H., *Bilderbogen und würfelspiel*, Edition Leipzig, Leipzig, 1981.

- Vollmann,T.E., Berry,W.L. and D.C. Whybark, *Manufacturing planning and control systems, 3d ed.*, Irwin, Homewood, Ill., 1992.

- Von Aretin,W., *Strategonon, versuch die kriegsführung durch ein spiel darzustellen* (Strategonon, attempt to represent warfare by a game), Dollfusz, Ansbach, 1830.

- Von Neumann,J. and O. Morgenstern, *Theory of games and economic behavior*, Princeton University Press, Princeton, 1947.

- Wanous,J.P. and E.E. Lawler, "Measurement and meaning of job satisfaction", *Journal of Applied Psychology*, Vol 56 No 2 (April 1972), p 95-105.

- Weldon,J.-L., "Information system design and development", *Journal of MIS*, Vol 1 No 3 (Winter 1984-85), p 3-4.

- Weldon,L. and C.B. Mills, "Reading text from computer screens", *ACM Computing Surveys*, Vol 19 No 4 (December 1987), p 329-358.

- Wesselius,J.H., *Software quality control and software requirement specification*, Ph.D. Thesis, Technical University Delft, 1993.

- Westrup,C., "Information systems methodologies in use", *Journal of Information Technology*, Vol 8 (1993), p 267-275.

- Whang,S., "Contracting for software development", *Management Science*, Vol 38 No 3 (March 1992), p 307-324.

- Wijers,G.M., *Modelling support in information systems development,* Thesis publishers, Amsterdam, 1991.

- Winters,A. and E. Hendrix, *Het reduceren van de simulatie in een produktie-voorraad systeem* (Simulation reduction in a production-stock system), Unpublished research memorandum, Tilburg University, Tilburg, 1986.

- Wirth,N., "The programming language pascal", *Acta Informatica*, Vol 1 No 1 (1970), p 35-63.

- Wirth,N., "From programming language design to computer construction", *Communications of the ACM*, Vol 28 No 2 (February 1985), p 160-164.

- Wirth,N., *Algorithms + data structures = programs,* Prentice Hall, Englewood Cliffs NJ, 1976.

- Wolfe,J. and C.R. Roberts, "The external validity of a business management game, a five-year longitudinal study", *Simulation and Games*, Vol 17 No 1 (March 1986), p 45-59.

- Woltjer,G.B., *Coordination in a macroeconomic game, its design and its role in education and experiments*, Universitaire Pers Maastricht, Maastricht, 1995.

- Yeo,G.K. and F.H. Nah, "A participants' DSS for a management game with a DSS generator", *Simulation and Gaming*, Vol 23 No 5 (September 1992), p 341-353.

- Yourdon,E., *How to manage structured programming*, Yourdon, New York, 1976.

- Yourdon,E., *Decline and fall of the american programmer*, Yourdon Press, Englewood Cliffs NJ, 1992.

- Zimmermann,H.J. and Sovereign,M.G., *Quantitative models for production management,* Prentice Hall, Englewood Cliffs NJ, 1974.

Index

Curriculum Vitae

Rommert Jan Casimir was born in Leiden on June 28, 1938[21]. During the war, his family moved to Eindhoven, where he finished the secondary school (gymnasium) in 1956. Inspired by the then popular notion that economic policy could produce full employment and general welfare, he started to study economics at the University of Amsterdam. In 1963 he took a job as a programmer with Electrologica, where he designed part of an operating system for the EL-X8 which has left no marks in history because, in a time when quality still counted, it was definitely inferior to the THE system for that machine. In 1968 he returned to the university as a student in business economics at the Netherlands School of Economics (now Erasmus University) in Rotterdam, and a student-assistant to Euwe, the chess grandmaster who taught information systems and computer science at two universities until he reached 70. The choice for business instead of economics and Rotterdam instead of Amsterdam prevented Rommert both from involvement in the student uproars of the time and from becoming either a disillusioned Marxist or an economist with fashionable right-wing attitudes. Because of the prevailing scarcity of computer specialists he almost automatically was appointed as assistant professor in computer science after graduation in 1970. When Euwe was succeeded by Verhoeff, Rommert followed his advice never to publish anything unless it was really important, a status which Verhoeff accorded to very few papers. Instead of engaging in serious research, Rommert maintained minicomputer software, developed a time-sharing management game and designed server software to connect first generation PC's to a mini. When he finally submitted a paper to a computer journal, it was rightly rejected, and he concluded that as a non-mathematician, he was definitely at the wrong side of the ever-widening gap between computer science and information systems. In 1985, he seized the opportunity to do research on management gaming at the information systems department of Tilburg University, where he also taught courses and supervised masters theses on a broad range of information systems subjects.

Since 1969, Rommert is married to Tine de Graaf, a nurse and teacher of nursing who really deserves a biography in her own right. They have three daughters. Sytske just obtained her masters degree in English with a thesis on the horse in medieval English literature, Esther is now finishing her theatrical science studies and Tanja is a first-year student in country management.

---

[21] For the family history and the name see Casimir,H.B.G., *Haphazard reality: half a century of science*, Harper and Row, New York, 1983.

VOORWOORD

Toen ik het manuscript van dit proefschrift af had moest iets opzoeken over de stand van zaken in de informatica en informatiekunde in 1970. Zo kwam ik in het nummer van augustus 1970 van de *Communications of the ACM* een ingezonden brief van Peter Wegner tegen, waarin hij stelde dat misschien wel de enige manier om goed onderzoek te doen was om met opzet slechte artikelen te schrijven die zeker afgekeurd zouden worden, waardoor je aan de verleiding van het succes zou ontkomen. Ik heb deze raad opgevolgd en 25 jaar gewerkt aan het onderzoek dat ik leuk vond. Daarbij heb ik modieuze onderwerpen altijd kunnen vermijden.

Mijn eerste bedrijfsspel ontwierp ik in 1970 als afstudeerscriptie voor het doctoraal bedrijfseconomie aan de Nederlandse Economische Hogeschool. Als docent informatica aan de Erasmus Universiteit werkte aan bedrijfsspelen als toepassing van samenwerkende processen, in de jaren zeventig een belangrijk onderwerp in de theorie van de *operating systems*.

Ik ging me nog intensiever met bedrijfsspelen bezighouden toen ik aan de Katholieke Universiteit Brabant ging werken in een groot project dat het gebruik van computers in het onderwijs moest bevorderen. Na een jaar of twee ging het project in rook op. Mijn eigen onderzoek werd nog enige tijd als voorwaardelijk gefinancierd onderzoek voortgezet, maar het werd eind 1988 gestopt, net toen de eerste versie van Infogame gereed was.

Twee gebeurtenissen waren aanleiding om dit onderzoek weer op te vatten. Aart de Zeeuw, de toenmalige decaan van de economische faculteit, gaf te kennen dat een niet gepromoveerde docent weinig goeds te wachten stond. In plaats van te wijzen op de ijzersterke rechtspositie van het universitair personeel, nam ik de uitdaging liever aan. Belangrijker was dat Cees Takkenberg bereid was als promotor op te treden en samen met Martin Smits ook het initiatief nam om Infogame in het nieuwe eerstejaarsvak *Grondslagen Informatie-Kunde* op te nemen, waardoor ik de nodige experimentele gegevens kreeg. Toch zou ik dit werk niet hebben afgemaakt zonder de steun van mijn vrouw Tine, waarschijnlijk de enige die altijd is blijven geloven dat dit proefschrift ooit af zou komen, en mijn dochters Sytske, Esther en Tanja, die het woord *proefschrift* al kennen zolang ze kunnen praten.

Rolf Bijl had de leiding bij het gebruik van Infogame. Hij werd bijgestaan door de studentassistenten Anja Bakker, Selma van Hoorn, Michel Jongen, Frodo Lammers, Carmen Merckx, Ester Rijnders, Jeroen van Stratum, Annelies Straub en Helga van de Wijngaart. Ik kan niet alle studenten noemen die hebben meegespeeld, maar ik zal me hun enthousiasme herinneren. Rolf Bijl en Martin Smits namen ook het initiatief tot het gebruik van Infogame in twee andere colleges.

Alle leden van de vakgroep BIKA verdienen mijn dank voor het werk dat zij de laatste twee jaar van mij overgenomen hebben. Daarbij noem geen namen omdat ik zelfs niet weet wie wat gedaan heeft. Cees Takkenberg leidde mij met bekwame hand door het moeilijke terrein van de interpretatie van en verslaglegging over mijn onderzoeksresultaten en in het eindstadium droegen de commissieleden Prof.

Dr. J.L.A. Geurts, Prof. Dr. P.Th.M. Laagland, Prof. Dr. P.M.A. Ribbers en Prof. Dr. D.B.B. Rijsenbrij bij aan de laatste versie door aan te dringen op het schrappen van eenzijdige opvattingen en onduidelijke redeneringen.

Het ontbreken van een dankwoord aan programmeurs en typisten is noch aan het vergeetachtigheid, noch aan minachting voor hun werk te wijten Het is puur het gevolg van het feit dat iedere informatiekundige tegenwoordig zelf kan programmeren en typen. In plaats daarvan wil ik alle huidige en vroegere medewerkers van de automatiseringsunit van de FEW van de KUB bedanken voor de hardware en software die ze mij ter beschikking hebben gesteld, en in het bijzonder voor het in de lucht houden van WP 5.1 voor DOS tijdens het schrijven van dit proefschrift. Door het gebruik van WP heb ik ook een index op kunnen stellen, die zelfs de namen van tweede en derde auteurs bevat.

Ook wil ik de medewerkers van de Universiteitsbibliotheek van de KUB bedanken voor het instandhouden van een schitterende collectie en een voortreffelijke informatievoorziening. Zonder *Excerpta Informatica* zou het verzamelen van de literatuur, met name voor hoofdstuk 2, een hopeloze zaak geweest zijn. Ook hierbij bedank ik niemand persoonlijk omdat zo veel werk achter de schermen gedaan is.

Tenslotte moet ik mijn Nederlandse lezers nog uitleggen waarom dit proefschrift in het Engels geschreven is. De reden daarvoor is eenvoudig dat ik hoop dat het buiten Nederland gelezen wordt. Ook de handleidingen van Infogame zijn, met het oog op gebruik in het buitenland, altijd in het Engels geschreven.

1 Methode van onderzoek

Jaarlijks worden er tientallen boeken en honderden artikelen over de ontwikkeling van informatiesystemen geschreven. Meestal worden daarin nieuwere, betere methoden beschreven. Onderzoek naar de kwaliteit van nieuwe en bestaande methoden wordt minder vaak gedaan.

Om na te gaan waarom dat zo is moeten we ons verdiepen in de verschillende *onderzoeksstrategieën*. Hoewel wetenschappers het niet over alle details eens zijn, bestaat er in grote lijnen eenstemmigheid over de indeling. Op de eerste plaats onderscheidt men *empirisch* en *niet-empirisch* onderzoek. Het niet-empirisch onderzoek bestaat uit *ontwerponderzoek* en *theoretisch onderzoek*, waarbij het laatste meer of minder formeel kan zijn. Vaak rekent men het formele onderzoek tot de informatica en het niet-formele onderzoek tot de informatiekunde.

Voor dit proefschrift is vooral het empirisch onderzoek van belang en dit kan naar twee gezichtspunten worden ingedeeld. Kijken we naar de plaats van het onderzoek dan kunnen we een onderscheid maken tussen *laboratoriumonderzoek* en *veldonderzoek*, kijken we naar de mate waarin de onderzoeker de onderzoeksomgeving beheerst dan onderscheiden we *experimenten* en *studies*. Meestal wordt de omgeving in het laboratorium wel en in het veld niet beheerst, zodat de *veldstudie* tegenover het *laboratoriumexperiment* wordt geplaatst. De onderzoeken die in hoofdstuk 5 van dit proefschrift behandeld wordt zijn echter laboratoriumstudies. Daarbij gaan we er van uit dat in een informatiekundig laboratorium, net als in

een psychologisch laboratorium, de reacties van proefpersonen bestudeerd worden. Het is dus wat heel anders dan een computerlaboratorium, waar apparatuur en programmatuur beproefd worden.

De volgende vraag is waarom het laboratoriumonderzoek, dat in de natuurwetenschappen, de medische wetenschappen en de psychologie zo belangrijk is, nauwelijks wordt toegepast om de kwaliteit van methoden voor het ontwikkelen van informatiesystemen te testen. Vier mogelijke argumenten tegen laboratoriumonderzoek zijn (a) de hoge kwaliteit van het theoretisch en ontwerponderzoek, (b) dat nieuwe produkten zichzelf toch in de markt moeten bewijzen, (c) dat de resultaten niet geldig zijn omdat men niet met representatieve proefpersonen kan werken en (d) dat er geen geschikte onderzoeksinstrumenten zijn. Tegen (a) en (b) kan men de hoge kosten van mislukte informatiesystemen aanvoeren, tegen (c) kan men inbrengen dat er ook medische proeven op ratten worden genomen terwijl een rat minder op een mens lijkt dan een student op een manager en om (d) te ondervangen is het in hoofdstuk 4 beschreven spel Infogame ontworpen.

De hoofdvraag in dit proefschrift is:

Wat zijn de voordelen en beperkingen van laboratoriumonderzoek bij de bestudering van methoden en hulpmiddelen voor informatiesysteemontwikkeling?

Om de voordelen aan te tonen gaan we uit van een algemeen aanvaarde hypothese, namelijk:

Er bestaat een verband tussen de kwaliteit van opeenvolgende mijlpalen binnen de systeemontwikkelingscyclus

Met behulp van het nieuw ontworpen bedrijfsspel Infogame scheppen we een omgeving waarin deze vraag beantwoord wordt en we laten zien dat dit laboratorium ook gebruikt kan worden voor het bewijs van meer betwiste stellingen.

2 Informatiesystemen

Allereerst vragen we ons af welk karakter het vakgebied *ontwerp van informatiesystemen* nu eigenlijk heeft. Gaat het over het brede gebied van het gebruik van informatie in organisaties of alleen maar over het automatiseren van al bestaande gegevensverwerkende systemen, gaat het om ontwerp of om onderzoek en hoort het vakgebied informatiesystemen tot de wiskunde of de sociale wetenschappen. Op die manier kunnen we acht opvattingen over het vakgebied onderscheiden, waarbinnen soms ook nog verschillende stromingen bestaan. Het is dan ook geen wonder dat de beoefenaren van dit vak het werk van anderen soms op zijn zachtst gezegd onpraktisch of onwetenschappelijk noemen. Er is echter één ding waar bijna iedereen het over eens is, en dat is dat te veel informatiesystemen mislukken, als is het moeilijk daar harde cijfers over te vinden.

Als reactie hierop heeft men gezocht naar methoden om de kwaliteit van informatiesystemen te meten. Daarbij kijkt men aan de ene kant naar de tevredenheid van gebruikers als maatstaf voor de kwaliteit, aan de andere kant heeft men *scoringmodellen* opgesteld waarmee de kwaliteit

berekend wordt op grond van factoren die belangrijk geacht worden. Zie hiervoor het proefschrift van van der Pijl en de artikelen van Delen en Rijsenbrij.

Een andere stap was het verbeteren van de kwaliteit van informatiesystemen met behulp van gestructureerde methoden. In de jaren zeventig en tachtig nam de ontwikkeling van methoden als het Nederlandse SDM, het Zweedse ISAC en het Britse SSADM een hoge vlucht. Een artikelserie over deze methoden verscheen in februari 1981 tot maart 1982 in het maandblad *Informatie*.

Bij het gebruik van een gestructureerde ontwikkelmethode wordt een project verdeeld in modulen of fasen die met een mijlpaal worden afgesloten, terwijl tevens wordt geregeld wanneer aan een module wordt begonnen. Het voordeel van de toepassing van een gestructureerde methode kan zijn dat een project daardoor beter te beheersen is, maar ook dat zo een betere verdeling van werk en een grotere druk op het personeel bereikt wordt. Als we aan zouden nemen dat de laatste argumenten de doorslag geven dan zou het nut van gestructureerde methoden alleen in het veld onderzocht kunnen worden. In ons laboratoriumonderzoek bestuderen we slechts de voordelen van gestructureerde methoden die niet afhankelijk zijn van organisatorische en menselijke verhoudingen.

Belangrijke verschillen tussen systeemontwikkelingsmethoden zijn de manier waarop een project in modules wordt verdeeld, de volgorde van de modules die wordt aangehouden, de wijze waarop het project wordt beheerst en de talen en hulpmiddelen die bij de ontwikkeling worden gebruikt. Wat betreft de indeling in modules wordt doorgaans gekozen voor een verdeling in *fasen*. Terwijl vroeger meestal een lineaire volgorde werd gekozen waarbij de fasen één voor één werden afgewerkt, kiest men tegenwoordig vaker een iteratieve strategie, waarbij een eerdere fase nog eens wordt uitgevoerd. Prototyping kan als een voorbeeld van een iteratieve strategie worden beschouwd omdat de specificaties veranderd kunnen worden naar aanleiding van een test van het prototype. Vanuit de informatica hebben vooral de programmeertalen en hulpmiddelen veel aandacht gekregen. In de informatiekunde is de aandacht verschoven van systeemontwikkeling naar *strategische informatiesystemen*, *herontwerp van bedrijfsprocessen*, en het *uitbesteden* van de informatieverwerking. Daarnaast komt in de praktijk het gebruik van programmapakketten op.

3 Spelen en experimenten

We kunnen het experimenteel onderzoek bekijken vanuit het gezichtspunt van de ontwerper van hulpmiddelen voor experimenten en vanuit het gezichtspunt van de onderzoeker. De eerste bouwt een bepaald soort hulpmiddelen, bijvoorbeeld bedrijfsspelen, de tweede richt zich op een bepaald soort onderzoek, bijvoorbeeld het nut van kleurenschermen.

Een belangrijk hulpmiddel bij onderzoek is het spel (Zie "Bedrijfsspelen" in Koorevaar,P., Oonincx, J.A.M., and P. Ribbers: *Handboek bestuurlijke informatiekunde*, Samsom BedrijfsInformatie, Alphen aan den Rijn, 1995). Het spel heeft sinds eeuwen een belangrijke rol in de cultuur gespeeld. Vooral over kansspelen met kaarten en dobbelstenen is veel bekend

omdat ze herhaaldelijk door kerk en staat verboden werden. Een belangrijk onderscheid is dat tussen sporten, waarbij het aankomt op snelheid en behendigheid en strategische spelen, waarbij het op de juiste zet aankomt. De *speltheorie* voert de abstractie nog verder door: eerst wordt een spel als model van de werkelijkheid beschreven en daarna wordt dat spel wiskundig geanalyseerd.

Een belangrijk kenmerk van spelen is het abstractieniveau. We onderscheiden het schijnbare abstractieniveau, de mate van overeenkomst tussen het spelmateriaal en voorwerpen uit de werkelijkheid, en het intrinsieke abstractieniveau, de mate waarin de buitenwereld het spel beïnvloedt. Monopoly heeft een laag schijnbaar abstractieniveau (de kalverstraat bestaat echt), maar een hoog intrinsiek abstractieniveau (de stations leveren nog evenveel winst op als in de bloeitijd van de spoorwegen). Alleen woordspelletjes zoals Scrabble en quizspelletjes zoals Triviant hebben een laag intrinsiek abstractieniveau.

Bij computerspelen onderscheiden we het gebruik van de computer als tegenstander, een toepassing die meestal tot de kunstmatige intelligentie wordt gerekend, en het gebruik van de computer als hulpmiddel. In het laatste geval kan het net als bij de sport om snelheid (in *arcade games*) of om inzicht (in *adventure games* en *simulatiespelen*) gaan.

Bij *bedrijfsspelen* gaat het om inzicht, en daar komt nog bij dat de deelnemers in een bedrijfsspel met elkaar concurreren (anders spreken we van een simulatiespel) en dat een bedrijfsspel een laag intrinsiek abstractieniveau heeft. Kort gezegd is een bedrijfsspel immers een spel waarin de deelnemers op de stoel van de directie van een onderneming gaan zitten. Daarom nemen we ook aan dat spelers zich in een bedrijfsspel niet zoveel anders gedragen als managers in de werkelijkheid doen.

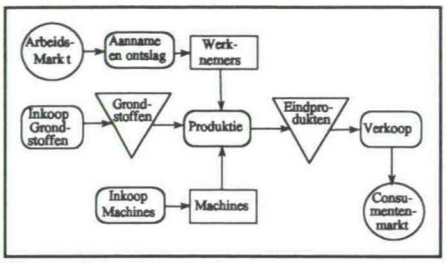
Bedrijfsspelen worden sinds het eind van de jaren vijftig gespeeld en zijn sinds die tijd niet fundamenteel veranderd. De oorlogsspelen die al in de negentiende eeuw gespeeld werden kunnen als voorlopers gezien worden.

Vooral in Amerika is veel experimenteel onderzoek naar de mens-computer interactie gedaan. Een voorbeeld is het onderzoek naar de vraag of grafieken of cijfers een betere ondersteuning voor beslissingen bieden. Dit onderzoek is in het begin van de jaren tachtig tamelijk populair geweest, maar het heeft geen eensluidende resultaten opgeleverd. Ook bleek het in de jaren twintig al eens uitgevoerd te zijn. Nog minder bevredigend zijn de resultaten van het onderzoek naar de invloed van beslissingsondersteunende systemen op de resultaten van ondernemingen in bedrijfsspelen. Daarbij zijn echter geen spelen gebruikt die, zoals Infogame, speciaal met het oog op dit onderzoek zijn ontworpen.

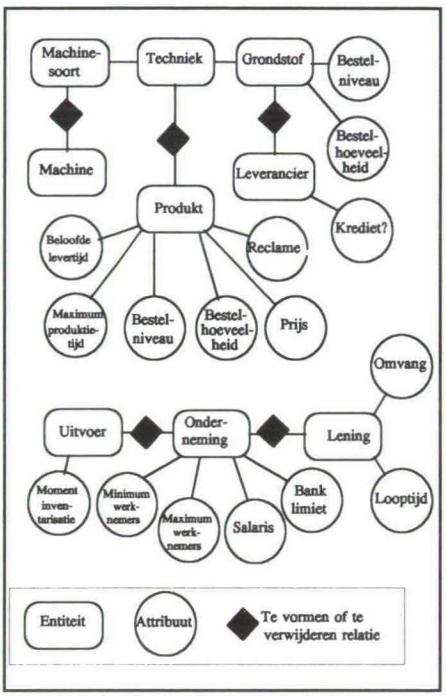
4 Infogame

Infogame is een door mij ontworpen bedrijfsspel voor onderwijs en onderzoek op het gebied van het ontwerp van informatiesystemen. Figuur 4.1 geeft een beeld van de materiaalstroom in de nagebootste ondernemingen, figuur 4.2 geeft een overzicht van de beslissingen die in Infogame genomen moeten worden.

Belangrijke verschillen met de gebruikelijke bedrijfsspelen zijn dat produktie en



Figuur 4.1: Materiaalstroom



Figuur 4.2: Entiteiten en relaties

verkoop in detail nagebootst worden en dat de spelers geen balans en verlies- en winstrekening of statistieken krijgen, maar alleen een bestand waarin alle gebeurtenissen in het afgelopen kwartaal, bijvoorbeeld de afzonderlijke verkooptransacties, zijn vermeld. Dat bestand is zo groot dat de spelers een informatiesysteem nodig hebben om Infogame goed te spelen.

Een ander kenmerk is dat produktie en verkoop worden geregeld door een *bestelniveau* en een *bestelhoeveelheid*. Als de voorraad beneden het bestelniveau komt wordt de bestelhoeveelheid geproduceerd of ingekocht.

Een uitgangspunt bij het ontwerp van Infogame was dat de spelers een beslissing alleen zouden hoeven te nemen als die een andere beslissing zou beïnvloeden. Dit geldt bijvoorbeeld voor het aantal personeelsleden, dat verband houdt met de geplande produktie. Een voorbeeld van een beslissing die niet samenhangt met andere beslissingen, en dus ook niet door de spelers in Infogame genomen hoeft te worden, is de verdeling van het reclamebudget over verschillende media. Tijdens het spelen is het ontwerp herhaaldelijk aangepast, onder andere omdat bepaalde regels moeilijk uit te leggen waren of omdat de spelers bepaalde gegevens misten. Het spel is daarmee eerder eenvoudiger dan ingewikkelder geworden.

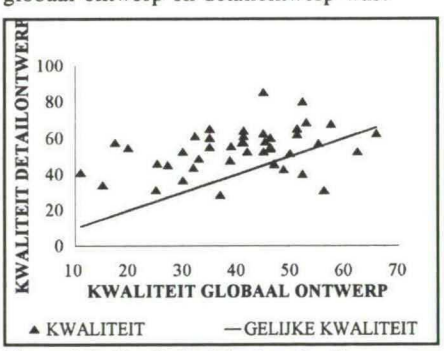
Een uitgebreide beschrijving van Infogame zelf is te vinden in de gebruikershandleidingen, die voor verschillende groepen spelers geschreven zijn.

5 Experimenten

Er zijn vijf soorten experimenten met vijf groepen studenten van de KUB uitgevoerd. Experiment 1, waarin studenten Bestuurlijke InformatieKunde (BIK) die deelnamen aan het college beslissingsondersteunende systemen naar eigen inzicht een informatiesysteem moesten ontwerpen, en experiment 5, waarin deze studenten een systeem moesten ontwerpen ten behoeve van studenten BedrijfsEconomie (BE)

die het spel speelden, hebben geen kwantitatieve gegevens opgeleverd. De belangrijkste resultaten komen uit experiment 2, waarin eerstejaarsstudenten BIK onder toezicht van studentassistenten op gestructureerde wijze een informatiesysteem hebben ontworpen en gebouwd. In experiment 3 hebben studenten BIK en BE de specificaties van informatiesystemen opgesteld en daarna het spel gespeeld met volgens die specificaties op maat gemaakte informatiesystemen en in experiment 4 hebben studenten BE eerst specificaties opgesteld en daarna het spel gespeeld met behulp van een standaardprogramma.

Uit experiment 2 (zie figuur 5.1) bleek duidelijk dat de kwaliteit van het detailontwerp gemiddeld beter was dan de kwaliteit van het globaal ontwerp en dat er ook een verband tussen de kwaliteit van globaal ontwerp en detailontwerp was.

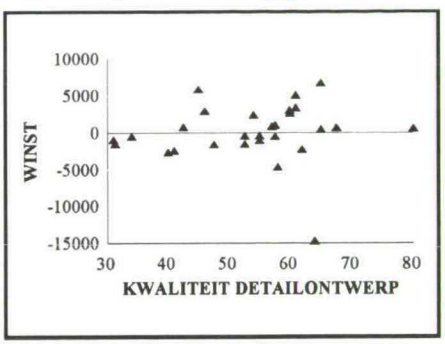


Figuur 5.1: Kwaliteit globaal en detailontwerp

Dit resultaat is niet schokkend, maar het geeft wel vertrouwen in Infogame als onderzoeksinstrument. Opmerkelijk was dat in het detailontwerp ook meer plaats was ingeruimd voor variabelen die bedrijfskennis vragen, zoals kostprijs en afschrijving.

Uit experiment 3 bleek het verwachte verband tussen de tevredenheid van de gebruikers en de lengte van het programma niet te bestaan en in experiment 4 bleek er geen verband te bestaan tussen de variabelen in de specificaties en de variabelen die in het standaardpakket opgevraagd werden.

In geen van de experimenten 2, 3 en 4 kon een verband tussen de kwaliteit van het informatiesysteem en de ondernemingswinst aangetoond worden. In figuur 5.2 is dit voor experiment 2 aangegeven.

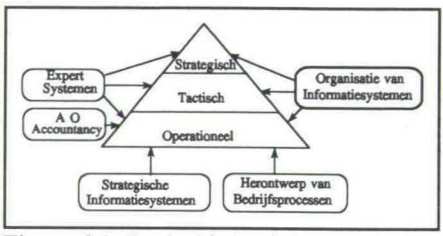


Figuur 5.2 Kwaliteit detailontwerp en winst

6 Conclusies

Bij een vergelijking van verschillende onderzoeksstrategieën blijken laboratoriumonderzoek enerzijds en theoretisch onderzoek en ontwerponderzoek anderzijds elkaar vooral aan te vullen. Veldonderzoek is een echt alternatief voor laboratoriumonderzoek. Om na te gaan welk soort onderzoek met Infogame gedaan kan worden gaan we uit van figuur 6.1. Hierin staan naast de traditionele onderwerpen uit de informatiekunde, namelijk het ontwikkelen van informatiesystemen op *strategisch*, *tactisch* en *operationeel niveau*, nog vijf nieuwere onderwerpen vermeld, namelijk de bouw van *expertsystemen*, *accountancy*

en administratieve organisatie, *strategische informatiesystemen*, *herontwerp van bedrijfsprocessen* en de *organisatie van informatiesystemen*, waarbij bepaald wordt op welk niveau beslissingen genomen worden en informatie wordt verzameld.



Figuur 6.1: Deelgebieden informatiekunde

Als daar een sponsor voor wordt gevonden is het goed mogelijk om de spelers expertsystemen te laten bouwen die binnen Infogame de produktie of marketing op tactisch niveau regelen. Ook kan Infogame zo worden veranderd dat er fouten optreden waardoor er behoefte aan de werkzaamheden van een (EDP-)accountant ontstaat.

Het ontwerp van strategische informatiesystemen en het herontwerp van bedrijfsprocessen vereisen echter dat de speler de organisatie van een in Infogame nagebootste onderneming kan veranderen en dat is nu niet mogelijk. De spelers kunnen ook geen keuze maken uit verschillende bedrijfsmodellen omdat ze dan te gauw geneigd zouden zijn de beroemde voorbeelden te volgen. Dit betekent dat we voor onderzoek op het gebied van deze onderwerpen voorlopig vooral op veldonderzoek aangewezen zijn.

Curriculum Vitae

Rommert Jan Casimir is geboren te Leiden op 28 juni 1938 (Zie voor de geschiedenis van de familie en de naam Casimir: H.B.G. Casimir: Het toeval van de werkelijkheid : een halve eeuw natuurkunde, Meulenhoff, Amsterdam, 1983). In de oorlog verhuisde hij met zijn ouders naar Eindhoven waar hij aan het Lorentz Lyceum in 1956 het diploma gymnasium behaalde. Aangemoedigd door de toen gangbare opvatting dat de economische politiek volledige werkgelegenheid en algemene welvaart zou kunnen brengen, ging hij economie studeren aan de Universiteit van Amsterdam. In 1963 ging hij als programmeur bij Electrologica werken. Een van zijn taken daar was de bouw van een deel van een operating systeem voor de EL-X8 dat geen plaats in de geschiedenis heeft gekregen omdat het duidelijk minder goed was dan het THE systeem voor dezelfde machine, en dat in een tijd waarin kwaliteit nog telde. In 1968 keerde hij terug naar de universiteit als student bedrijfseconomie aan de NEH (nu Erasmus Universiteit) in Rotterdam en student-assistent bij Euwe, de schaakgrootmeester die tot zijn zeventigste informatica en informatiekunde aan twee universiteiten doceerde. De keuze voor bedrijfseconomie in plaats van algemene economie en Rotterdam in plaats van Amsterdam voorkwam dat Rommert betrokken raakte in de toenmalige studentenopstanden en dat hij later een gedesillusioneerde marxist of een econoom met modieuze rechtse opvattingen zou worden. Als gevolg van de toenmalige schaarste aan computerspecialisten werd hij na zijn afstuderen bijna automatisch benoemd tot wetenschappelijk medewerker in de automatische informatieverwerking. Toen Euwe werd opgevolgd door Verhoeff volgde Rommert diens advies op om nooit iets te publiceren tenzij het echt belangrijk was, wat volgens Verhoeff niet gauw het geval was. In plaats van met serieus onderzoek hield Rommert zich bezig met het onderhoud van software voor minicomputers, de ontwikkeling van een time-sharing bedrijfsspel en het ontwerp van software voor een server die de eerste generatie PC's met een minicomputer verbond. Toen hij tenslotte een artikel naar een informaticatijdschrift opstuurde werd dat terecht geweigerd en zag hij in dat hij als niet-wiskundige kennelijk aan de verkeerde kant van de steeds wijder wordende kloof tussen informatiekunde en informatica zat. In 1985 nam hij dan ook de gelegenheid waar om onderzoek op het gebied van bedrijfsspelen te doen bij de vakgroep Bestuurlijke Informatiekunde van de Katholieke Universiteit Brabant, waar hij zich verder door het geven van colleges en het begeleiden van afstudeerders met een groot aantal onderwerpen op het gebied van de informatiekunde bezig hield.

Rommert is sinds 1969 getrouwd met Tine de Graaf, een verpleegkundige en leraar verpleegkunde die een biografie voor zich zelf verdient. Zij hebben drie dochters. Sytske heeft net haar doctoraal Engels afgelegd met een scriptie over het paard in de middeleeuwse Engelse literatuur, Esther zit in het laatste jaar van haar studie theaterwetenschappen en Tanja is eerstejaarsstudent landinrichtingswetenschappen.